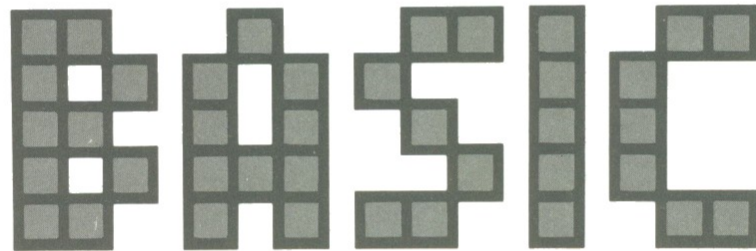
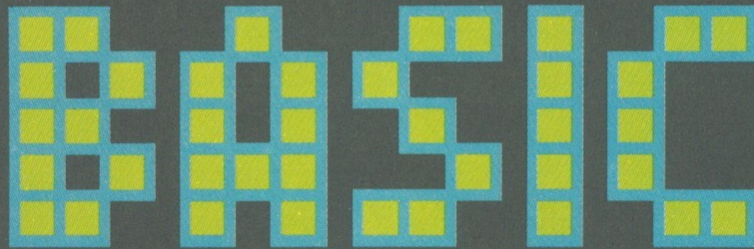
PROF. RENATO M. E. SABBATINI

INTRODUÇÃO À PROGRAMAÇÃO BASIC PARA MICROCOMPUTADORES

NOVA CULTURAL

PROF. RENATO M. E. SABBATINI

INTRODUÇÃO À PROGRAMAÇÃO BASIC PARA MICROCOMPUTADORES

NOVA CULTURAL

PROF. R. SABBATINI

BASIC

INTRODUÇÃO À PROGRAMAÇÃO BASIC PARA MICROCOMPUTADORES

CAPÍTULO 1

O QUE É UM COMPUTADOR?

Desde épocas imemoriais, o homem tem inventado ferramentas e instrumentos para ajudá-lo a fazer milhares de coisas diferentes. Podemos imaginar a ferramenta como uma espécie de extensão dos sentidos e dos músculos do ser humano. O que é um telescópio, por exemplo? Nada mais do que a ampliação do sentido da visão. Sem instrumentos e ferramentas, certamente o homem não teria atingido o grau de civilização que tem hoje, nem progrediria tão depressa.

Também o computador pode ser visto como uma espécie de ferramenta que o homem inventou para o ajudar, ampliando os poderes de seu próprio cérebro. É usando o seu cérebro que o homem é capaz de resolver problemas, decidir coisas e memorizar informações. Com a ajuda do computador, entretanto, ele pode fazer cálculos complicadíssimos muito mais rapidamente do que com outros meios.

O computador também tem MEMÓRIA, ou seja, ele é capaz de guardar as informações e reutilizá-las. Essas informações, que são chamadas de DADOS, devem ser colocadas por um ser humano na memória do computador.

Muitas vezes, os computadores são capazes de memorizar coisas que os homens não conseguem: listas longas de nomes e de números, por exemplo.

Um computador é capaz de achar rapidamente qualquer informação guardada em sua memória. Ele pode também comparar dados entre si, para mostrar, por exemplo, qual é o maior ou o menor.

Tudo isso faz do computador uma ferramenta extremamente poderosa, um auxiliar indispensável do homem moderno, em um número enorme de atividades, nos bancos, nas empresas fornecedoras de energia, nas telecomunicações, nas escolas etc.

Um computador pode fazer tudo?

Entretanto, mesmo sendo tão poderoso, o computador é uma máquina relativamente primitiva, quando comparada com o cérebro humano. O computador não tem sentimentos, nem iniciativa própria. Ele só faz o que um ser humano mandar: essa pessoa é chamada PROGRAMADOR, e sua função é colocar no computador uma série de comandos ou instruções, que dizem exatamente o

que a máquina deverá realizar. Esta série de ordens é chamada de PROGRAMA e, sem este, nenhum computador funciona.

O computador é uma ferramenta universal

A grande vantagem do computador sobre os outros instrumentos é que, com um só aparelho, podemos fazer muitas coisas diferentes. Para isso, basta colocar um outro programa em sua memória, e ele lhe obedecerá. Por isso, dizemos que o computador é uma ferramenta universal. Hoje em dia, com os robôs (que são computadores dotados de força e movimento), o número de coisas que se pode fazer é maior ainda. Portanto, sabendo programar um computador, você poderá dominá-lo completamente.

É importante aprender programação?

Em muitas profissões, o computador é uma ferramenta de trabalho muito importante e necessária. Quem quiser estudar para engenheiro civil, por exemplo, e não souber computação, vai ser incapaz de construir prédios grandes, estradas e túneis. Outras profissões, que ainda não usam muito o computador, usarão cada vez mais no futuro. Exemplos: Medicina, Biologia, Educação, Direito, etc. Por isso é importante que todos aprendam alguma coisa sobre os computadores.

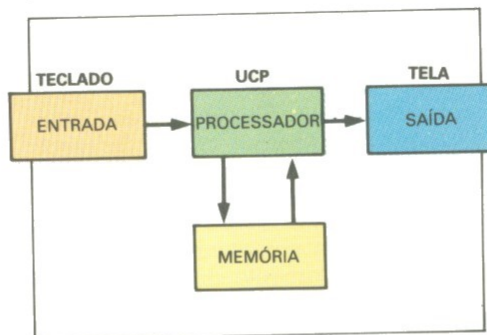
É claro que nem todas as pessoas que usam computadores precisam saber programá-los. Pode-se comprar um programa pronto, e bastará colocá-lo no computador e utilizá-lo devidamente. Entretanto, só se conseguirá dominar completamente o computador se se souber programá-lo. Muitas vezes, não se achará um programa feito por outros que realize exatamente o que se quer. E, além disso, como se sabe, a programação de computadores é uma profissão cada vez mais valorizada e atraente.

Até há quatro ou cinco anos a maioria das pessoas não tinha a menor chance de utilizar diretamente um computador no seu dia-a-dia. Entretanto, com o progresso maravilhoso da eletrônica, surgiram computadores cada vez menores e mais baratos, que hoje estão ao alcance de todos. Esses novos computadores, chamados de microcomputadores, estão entrando em todos os ramos da atividade humana, e em breve serão tão comuns quanto os telefones e os automóveis. Devemos nos preparar para esse futuro!

COMO É UM COMPUTADOR?

Para entender como é organizado um computador, temos antes que examinar a maneira como ele é utilizado.

Podemos imaginar o computador como uma espécie de calculadora. Para efetuar um cálculo por meio dela, precisamos primeiro fornecer-lhe os valores que serão utilizados nas contas. Isso é feito pelas teclas, que também são usadas para informar à calculadora quais serão as operações a realizar (soma, divisão etc.). Uma vez fornecidos os dados e as operações, a calculadora efetua os cálculos e mostra os resultados no visor.

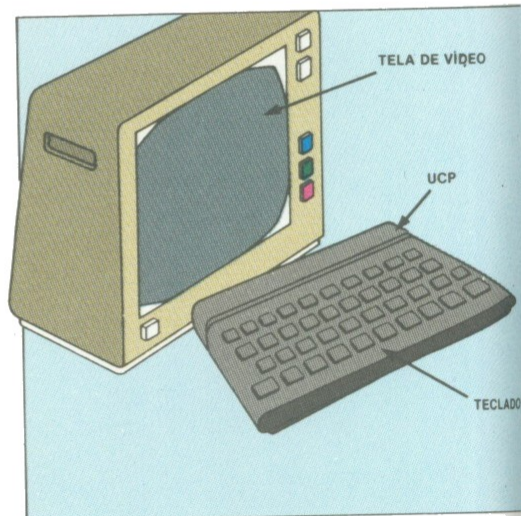
A forma de utilizar um computador não é muito diferente desta. Ele é formado por diversos elementos, cada um encarregado de uma função específica.

ENTRADA: é a parte do computador que recebe os DADOS (números, letras) e o PROGRAMA (ordens e operações).

PROCESSADOR: é o elemento do computador que mostra os resultados.

Um computador pode ter diferentes tipos de entradas e saídas, ao contrário das calculadoras mais simples. Na maioria dos computadores pequenos (chamados de microcomputadores) estas são as partes responsáveis pelas três tarefas:

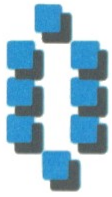
1) O TECLADO — parecido com o de uma máquina de escrever, é usado para informar ao computador os dados e as instruções ou comandos. Portanto, ele é o elemento de ENTRADA do computador.

2) A UNIDADE CENTRAL DE PROCESSAMENTO — esse é o nome da calculadora que o computador tem em seu interior. Na maioria dos mi-

crocomputadores, essa unidade está localizada na mesma caixa do teclado, e é formada por milhares de circuitos eletrônicos de tamanho microscópico. Essa parte é o PROCESSADOR do microcomputador, e é também conhecida pela abreviatura (UCP).

3) A TELA DE VÍDEO — a informação que for digitada (datilografada) no teclado aparece ao mesmo tempo na tela de uma TV, ligada ao computador: desse modo, pode-se ver o que se está fazendo. Quando o resultado estiver pronto, o computador também vai escrevê-lo na tela. Portanto, ela é o elemento de SAÍDA do nosso computador.

Outra parte importante do computador é a MEMÓRIA, da qual já falamos acima. A memória do computador é capaz de guardar ao mesmo tempo um certo número de informações. Por exemplo, muitos microcomputadores podem guardar até 48 000 números ou letras em sua memória. Por isso se diz que eles têm uma memória de 48 K (o K significa quilo, ou 1 000 vezes: portanto,

48 × 1 000 = 48 000).

Não se vê a memória, a não ser que se abra a caixa do teclado. Esta é formada por milhares de circuitos microscópicos, do mesmo tipo que os do processador.

O que é um programa de computador?

Os computadores recebem ordens em uma linguagem própria, diferente da nossa. Assim, você terá que aprendê-la, para conseguir se comunicar com ele.

A linguagem que a maioria dos microcomputadores é capaz de entender chama-se BASIC. Ela foi inventada nos Estados Unidos há cerca de vinte anos. Por isso, cada ordem ou comando na linguagem BASIC é uma palavra em inglês. Entretanto, o número desses comandos é pequeno, o que torna fácil decorá-los. Com algumas aulas, você já saberá usá-los como se fossem sua própria linguagem. Aprender BASIC é muito mais fácil que aprender qualquer idioma estrangeiro.

Eis um exemplo de comando simples em BASIC:

PRINT 8 + 3

Esta ordem ou comando do programa tem o seguinte significado para o computador:

ESCREVA O RESULTADO
DA CONTA 8 + 3

A palavra *print* quer dizer, "escreva", em inglês.

Normalmente um programa é constituído de uma série de instruções ou comandos dados ao computador. Ele realiza um de cada vez, na ordem em que eles forem fornecidos.

Todo programa digitado no teclado do computador é, primeiramente, guardado em sua memória. Assim ele pode se lembrar do programa, e repeti-lo para nós, tantas vezes quantas quisermos. Essa é sua grande diferença em relação às calculadoras.

Infelizmente, a memória do computador se apaga ao desligarmos a força. Todas as informações guardadas se perdem, inclusive o programa.

Por esse motivo, seria bom ter uma maneira de guardar o programa em outro lugar, para que não se perca ao desligarmos o computador. Assim não será necessário digitar de novo todo o programa, ao ligarmos o computador.

Os microcomputadores têm uma maneira muito simples de guardar esses programas: um gravador cassete comum. Através de um fio que liga o computador ao gravador, todas as instruções de um programa podem

TESTE

1.1. Marque V se for verdadeiro, ou F se for falso:

( ) O computador é capaz de pensar sozinho.
( ) O computador fica chateado se errar algum cálculo.
( ) O computador não erra quando realiza algum cálculo.
( ) O computador não tem memória.
( ) O computador faz contas mais rapidamente que o homem.
( ) O computador pode fazer coisas sem ser programado pelo homem.

1.2. Defina com suas próprias palavras:

PROGRAMA:
DADOS:

1.3. Diga se concorda ou não com a seguinte afirmação, e por quê:

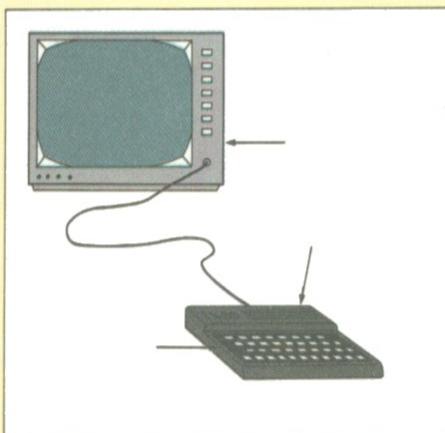
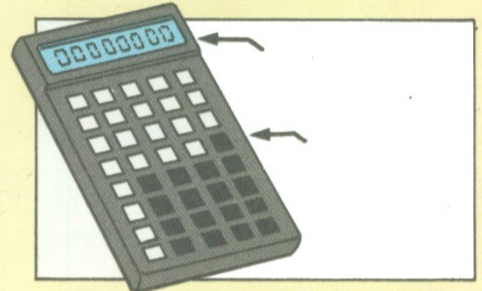
UM DIA OS COMPUTADORES VÃO MANDAR NA HUMANIDADE

1.4. Cite três trabalhos que o homem faz com auxílio dos computadores:

1 -

2 -

3 -

1.5. Escreva, na figura abaixo, os nomes das partes do microcomputador:

1.6. Uma calculadora eletrônica comum é uma espécie de computador simplificado. Escreva na figura abaixo quais são os elementos de ENTRADA e de SAÍDA da calculadora.

1.7. Qual a diferença entre o teclado de um computador e o de uma calculadora?

1.8. Escreva para que serve cada parte do computador:

TECLADO:
TELA DE VÍDEO:
UNIDADE DE PROCESSAMENTO:
MEMÓRIA:

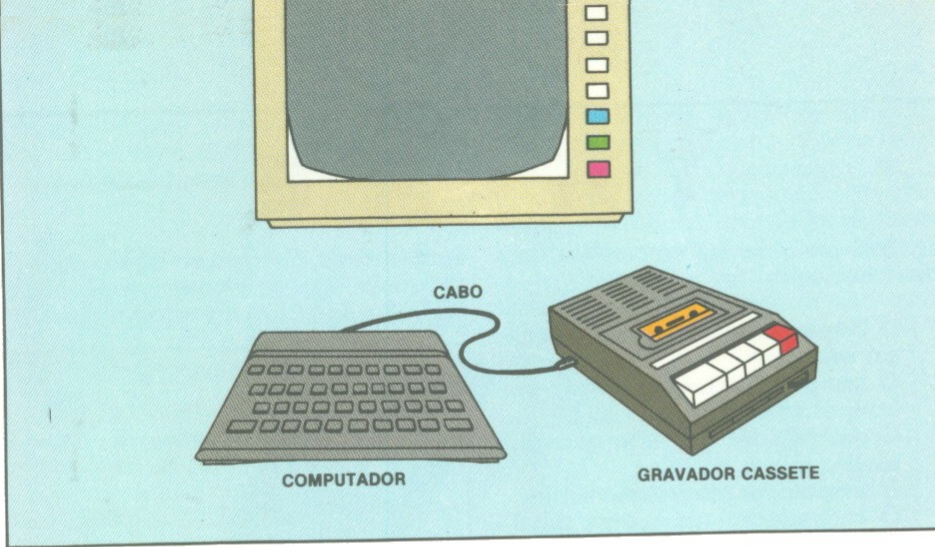
1.9. O que é memória auxiliar, e qual é o tipo de memória auxiliar que os microcomputadores mais comuns utilizam?

1.10. Marque V se for verdadeiro, F se for falso:

( ) BASIC é uma linguagem de computador.
( ) Um idioma estrangeiro é mais simples do que o BASIC.
( ) O BASIC é uma linguagem inventada pelo homem.
( ) Um computador só pode ser programado na sua própria linguagem.
( ) Um programa é um conjunto de instruções ao computador.
( ) O programa de computador precisa ser guardado na memória.
( ) Existe uma parte da memória do computador que não pode ser apagada.

1.11. Explique o que significam para o computador os seguintes comandos em BASIC:

PRINT 12 − 45
PRINT 29 + 3 − 8

1.12. Por que o microcomputador entende BASIC logo que você o liga?

ser gravadas em uma fita cassete, quando você quiser.

O computador grava essas informações usando uma espécie de código Morse de alta velocidade. Quando você quiser colocar de novo o programa na memória do computador, basta "tocar" para ele a fita gravada, e ele reconhecerá o programa, automaticamente.

Assim, o gravador auxilia a memória principal do computador e por isso ele é chamado de "memória auxiliar".

À medida que você aprender BASIC, você também será capaz de entender os programas de computador escritos nessa linguagem. E poderá também começar a escrever outros programas — os seus programas!

Quando você ligar o microcomputador, ele já estará pronto para aceitar instruções em BASIC, digitadas por você no teclado.

Ele sabe BASIC porque essa linguagem já vem gravada em uma memória especial, colocada na fábrica, dentro da caixa do teclado. Essa memória nunca se apaga, ao contrário da outra, que vai ser usada para colocar os seus programas e dados.

COMO ESTUDAR BASIC

O objetivo deste curso é ensinar você a programar microcomputadores usando a linguagem BASIC. Praticamente todos os microcomputadores hoje existentes são capazes de entender essa linguagem: assim, o que você aprender aqui poderá ser aproveitado para vários tipos e marcas de computadores pequenos ou mesmo grandes.

Entretanto, existem muitas variantes da linguagem BASIC, conforme a marca do computador: alguns comandos que existem em um computador não existem em outro, ou então funcionam de modo diferente. Essas variantes são chamadas de "dialetos". Existem, por exemplo, dialetos diferentes para os micros tipo TK-85, CP-500, TK-2000 etc.

Em que computador estudar?

Para que você aprenda melhor, sem ficar confuso com os diversos dialetos existentes, este curso ensinará apenas a variante do BASIC para os microcomputadores menores e mais baratos: TK-82C, TK-83, CP-200 e Ringo Essas são marcas nacionais de microcomputadores, que falam exatamente a mesma linguagem BASIC, e que são similares (compatíveis) com o microcomputador inglês chamado Sinclair ZX-81 e o norte-americano Timex 100.

Tudo o que você aprende aqui se aplica a todos os computadores dessa linha, mas não a computadores de outras marcas.

Diferenças entre os computadores

Existem pequenas diferenças entre as diversas marcas indicadas. Neste curso não ensinaremos como montar ou ligar cada tipo de computador: isso é facilmente encontrado nos manuais que acompanham os micros. Quanto à operação do teclado e do vídeo, as diferenças são as seguintes:

— os computadores da Microdigital (TK-82C, TK-83 e TK-85) têm exatamente o mesmo sistema de operação e disposição das teclas. A única diferença é que o TK-82C e o TK-83 têm um teclado de membrana (sem partes móveis), ao passo que o TK-85 têm teclas mecânicas;

— a tecla chamada de NEWLINE, nos computadores tipo TK, é chamada de ENTER, nos computadores CP-200 e Ringo. Entretanto sua função é exatamente igual em todos;

— a disposição do teclado CP-200 e do Ringo é ligeiramente diferente da encontrada nos TK. O CP-200 tem duas teclas SHIFT, ao invés de uma, e tem duas teclas RESET vermelhas, na parte superior do teclado. O Ringo tem uma tecla de ENTER em uma disposição diferente dos anteriores, marcada com uma flecha para baixo;

— a forma como aparecem textos e gráficos na tela de TV também pode ser diferente. Nos computadores TK, as letras são escuras, sobre fundo branco. No CP-200, normalmente é o contrário (letras brancas sobre fundo escuro), mas isso pode ser facilmente invertido. No Ringo, você pode mudar de uma cor para outra com o auxílio de um interruptor.

Quanto ao restante da operação, todos os computadores funcionam da mesma maneira.

Um aviso importante: embora seja possível aprender programação sem computador, isso não é recomendável. Este curso permite que você teste as lições e os programas e realize os exercícios e os testes em um computador qualquer, do tipo indicado aqui. Você verá que o seu rendimento será muito maior.

Como utilizar este curso

As lições do curso de BASIC são organizadas em capítulos, que irão apresentando os comandos e sua utilização em uma certa ordem. A não ser que você já conheça BASIC, não tente pular capítulos ou estudar fora da ordem. Os capítulos são divididos em seções distintas, cada um com três partes:

— texto
— exercícios
— teste

Você deve inicialmente ler o texto, seguindo atentamente todos os exemplos. Teste os exemplos dados no computador e veja os resultados. Tente modificar os programas, em seguida realize os exercícios propostos. Use sua imaginação e tente fazer programas, usando sua própria cabeça.

Finalmente, realize o teste e veja o seu desempenho. Se você errou muito ou não conseguiu responder a todas as questões, estude a seção mais uma vez.

E lembre-se: não adianta aprender uma linguagem qualquer, para depois não usá-la. Fatalmente você a esquecerá após um certo tempo. Durante e após o curso, use bastante o computador, sempre tentando criar novos programas!

COMO USAR O MICROCOMPUTADOR?

Agora que você já sabe como é o microcomputador, vamos aprender a operá-lo.

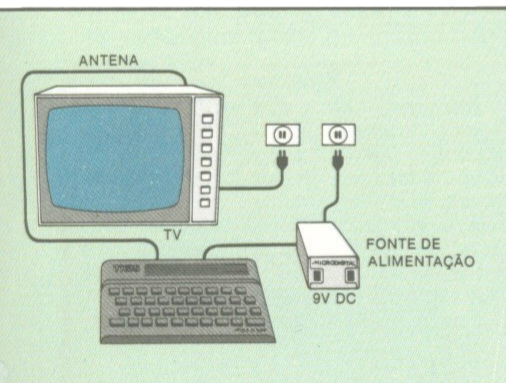
Montagem do microcomputador

As instruções completas para ligação do computador podem ser lidas no próprio *Manual* do equipamento. Aqui damos apenas algumas instruções gerais.

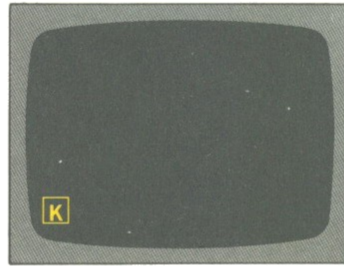
1) Ligue primeiro o computador, por meio de um cabo apropriado, no lugar onde se liga a antena externa, no aparelho de TV. Se algum fio já estiver ligado nesse lugar deve ser retirado.

2) Agora ligue o computador na tomada. Nos computadores tipo TK, isto deve ser feito através de uma FONTE DE ALIMENTAÇÃO.

3) Agora ligue o fio da TV em uma outra tomada.

Veja como deverão estar ligados os fios:

4) Agora ligue o aparelho de TV. Coloque o seletor de canais no canal 2.

5) Para ligar o computador use o interruptor marcado (LIGA-DESLIGA).

6) Gire o botão de sintonia até obter a imagem de uma letra K na parte inferior da tela.

7) Usando os botões de brilho e de contraste da TV, ajuste-os até que a letra K apareça o mais nitidamente possível na tela.

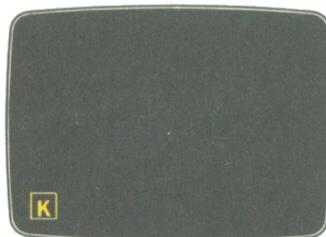
Se você não conseguir enxergar bem a letra K, aperte algumas teclas do teclado. Veja como as letras que você apertou vão aparecendo na tela. Assim ficará mais fácil acertar a sintonia.

Se o computador já estiver montado, é muito mais simples colocá-lo em funcionamento:

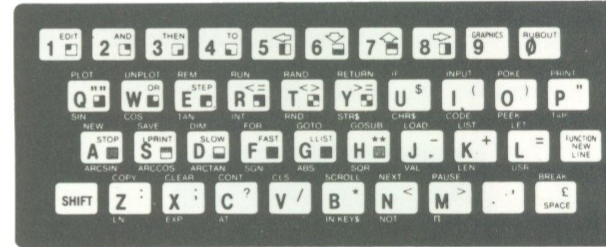
1) ligue o aparelho de TV.
2) ligue o computador.

Para desligar o computador, faça exatamente ao contrário: primeiro desligue o botão LIGA-DESLIGA, localizado atrás da caixinha, depois desligue o botão de força do aparelho de TV.

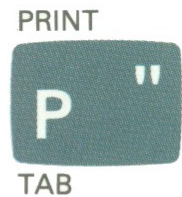
Experimente agora ligar e desligar o computador uma, duas ou três vezes. Não fique com medo de danificar o computador com esse procedimento: ele é feito para suportar bem essas operações. Inicialmente não vamos usar o gravador cassete, por isso não é necessário aprender agora como ele é ligado ao computador.

A primeira coisa que aparece na tela da TV, quando você liga o computador, é a letra K, dentro de um quadrinho preto, na parte de baixo da tela:

Este quadrinho é chamado de CURSOR. Toda vez que ele aparecer na tela, no mesmo lugar, significa que o computador está pronto para aceitar suas ordens ou comandos, em linguagem BASIC.

PARA DAR COMANDOS AO COMPUTADOR USAMOS O TECLADO

Olhe agora para o teclado.

1) As teclas com letras e números têm a mesma disposição que as de uma máquina de escrever comum: Q, W, E, R, T, Y, etc... Letras, algarismos ou sinais de pontuação estão escritos numa mesma tecla.

Ao conjunto de todas as letras e sinais gravados nas teclas chamamos de CARACTERES DE TEXTO.

2) Em cada tecla existem vários símbolos ou palavras, que são chamados RÓTULOS DA TECLA. Algumas teclas têm apenas 2 símbolos ou palavras, outras têm até 6. Olhe a tecla da letra P:

Você pode ver que o símbolo mais em cima de todos é PRINT.

Logo abaixo dele, temos um sinal em vermelho: aspas.

Finalmente, embaixo de todas, vemos a palavra TAB.

Todos estes símbolos e palavras podem ser introduzidos (informados) ao computador, usando-se a mesma tecla: a tecla da letra P.

Mas como o computador sabe qual palavra ou símbolo nós vamos introduzir, se a tecla é uma só?

Isto é muito simples. Vamos experimentar um pouco para você perceber como é feito.

A postos? Muito bem:

Pressione a tecla P uma vez. Não precisa apertar muito forte. Note que se ouve um sinal sonoro fininho quando se aperta a tecla. Ele é chamado de bip.

Quando o computador soar o bip, quer dizer que ele aceitou aquilo que você quis transmitir a ele, ao pressionar a tecla.

Olhe para a tecla agora:

Veja como na parte de baixo apareceu a palavra PRINT. O cursor agora

foi deslocado para depois da palavra PRINT na tela, e ainda é um quadradinho preto.

Mas note que agora a letra dentro dele é L.

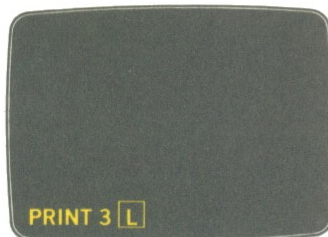
ESTA É A PRIMEIRA FUNÇÃO DA TECLA:

Quando o cursor na tela tem a letra K dentro, ao pressionarmos a tecla, a palavra ou símbolo que estiver bem em cima das outras é informada ao computador. Esta palavra é uma instrução ou comando ao computador, na linguagem BASIC.

Estamos dando ao computador uma ordem: ESCREVA.

Mas escrever o quê? Precisamos dar a ele alguma outra informação.

Pressione agora a tecla onde está o número 3.

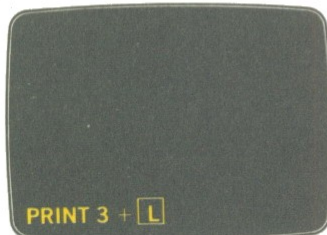
Veja o que ocorreu:

O algarismo 3 apareceu na tela, depois da palavra PRINT e um espaço em branco. O cursor mudou de novo para um lugar logo após o número 3, e ainda tem a letra L dentro.

Por isso que ele se chama CURSOR: ele mostra exatamente o ponto onde vai ser colocada mais uma outra informação, se você pressionar uma tecla qualquer.

ESTA É A SEGUNDA FUNÇÃO DA TECLA:

Quando o cursor na tela está com a letra L dentro, ao pressionarmos uma tecla, a letra ou número marcado são informados ao computador.

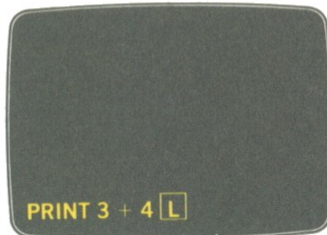
Agora pressione a tecla SHIFT, e pressione ao mesmo tempo a tecla K (podem existir duas teclas marcadas SHIFT: uma de cada lado do teclado, na fileira de baixo).

Veja como está a tela agora:

Apareceu o sinal + (mais), logo depois do número 3, certo? Note que o cursor mudou de novo de lugar, indo para depois desse sinal de +.

ESTA É A TERCEIRA FUNÇÃO DA TECLA:

Quando o cursor na tela tem a letra L dentro, ao pressionarmos uma tecla, ao mesmo tempo que a letra SHIFT, a palavra, ou símbolo, localizada em segundo lugar (em vermelho) é informada ao computador. Esta palavra também é um comando ou instrução em BASIC, mas, dependendo da tecla pode ser também um sinal qualquer.

Neste caso, o sinal + indica ao computador uma operação matemática: a soma.

Agora pressione a tecla marcada 4:

Na tela apareceu o número 4, logo depois do sinal de soma.

Estamos dando uma ordem ao computador:

PRINT 3 + 4

que em português significa:

ESCREVA O RESULTADO DE 3 MAIS 4

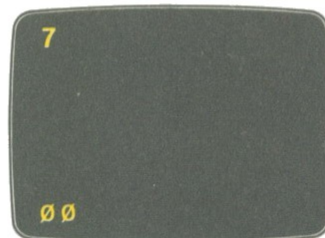
Só que o computador ainda não obedeceu à nossa ordem. Por quê?

Acontece que ele ainda não sabe se nós terminamos de digitar o comando completamente. No exemplo que estamos estudando, pode ser que eu quisesse somar outros números ao 3 ou 4, certo?

Para informarmos ao computador que o nosso comando já está completo e pode ser obedecido, pressionamos a tecla

NEW LINE.

Esta tecla significa *introduzir*, em português. Com isso, estamos introduzindo o comando que digitamos antes, e que está escrito na parte de baixo da tela, no computador. E ele poderá obedecer.

Pressione a tecla NEW LINE; veja o que acontece:

A tela fica toda preta, e depois de uma pausa bem pequena (o computador está calculando o resultado) aparece no alto da tela o número 7.

O número 7 é o resultado da operação 3 + 4, efetuada pelo computador, e ele é escrito na tela com o comando que você introduziu pelo teclado: PRINT.

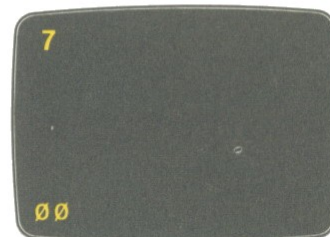
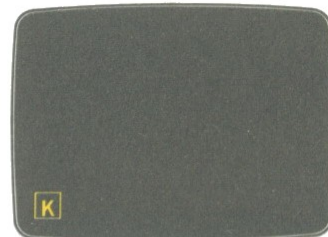
Na parte de baixo aparecem os números 0/0.

Isto significa que o computador não achou nenhum erro no comando que você introduziu. Isso às vezes pode acontecer!

Mais tarde vamos explicar direitinho o que significa isso.

Se você digitou errado alguma coisa na tela, e não sabe como apagá-la, pode desligar o computador. Torne a ligá-lo após alguns segundos, e comece tudo de novo.

Não se preocupe que isso não vai prejudicar o computador, nem mudar os ajustes já feitos na TV.

Após terminarmos o último exercício a tela estava assim:

Para voltar novamente o cursor K à tela, pressione a tecla NEW LINE:

O computador está pronto para aceitar outros comandos.

Resumo

Até agora vimos que existem dois tipos de cursores: K e L

K indica que o computador aceitará uma ordem ou comando, quando for pressionada uma tecla.

A instrução ou comando são palavras da linguagem BASIC, marcadas na parte superior do teclado, acima de cada tecla.

Exemplos de instruções em BASIC:

EDIT	STOP
PRINT	CLS

Localize cada uma destas palavras no teclado.

Se a palavra desejada estiver escrita em vermelho, a tecla SHIFT deverá ser pressionada ao mesmo tempo.

L indica que o computador aceitará uma letra, número ou sinal marcados em vermelho, a tecla SHIFT deverá ser pressionada ao mesmo tempo.

Como corrigir erros de digitação

Os erros que surgirem durante a digitação de uma linha podem ser corrigidos antes que pressionemos a tecla NEW LINE.

Para isso, usamos a tecla sobre a qual está escrito RUBOUT, no canto superior direito do teclado. RUBOUT quer dizer CANCELAR, em português.

Para usar a tecla RUBOUT, devemos pressionar ao mesmo tempo a tecla SHIFT, e a tecla 0 (zero). Note que a palavra RUBOUT está marcada em vermelho acima da tecla com o número 0. Por isso temos que usar a tecla SHIFT.

Cuidado! Se você esquecer de apertar a tecla SHIFT, fará aparecer na tela o número 0, ao invés de cancelar.

Vamos seguir um exemplo:

Vamos supor que você quer digitar o seguinte comando:

PRINT 3 + 25

Mas errou, e digitou:

PRINT 4 + 25

Introduza esse comando pelo teclado:

O cursor na tela está marcado [K].

Pressione a tecla P: aparece PRINT e o cursor muda para [L].

Pressione a tecla 4: aparece o número 4.

Pressione a tecla K juntamente com SHIFT: aparece o sinal +

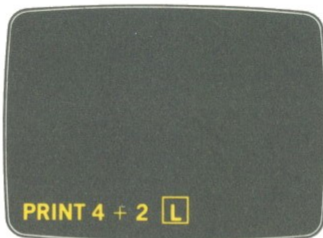
Pressione as teclas 2 e 5: aparece o número 25.

Agora, queremos corrigir o número 4, mudando-o para 3.

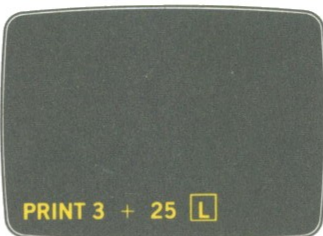
Existem duas maneiras de se fazer isso:

PRIMEIRA MANEIRA DE CORRIGIR UMA LINHA

Pressione a tecla SHIFT, e a mantenha abaixada com o dedo.

Agora pressione a tecla RUBOUT uma vez. Veja que o cursor L volta para trás, e apaga da tela o número 5.

Repita a operação (SHIFT e RUBOUT) por mais três vezes.

Veja que fica na tela somente o comando PRINT que você tinha digitado:

Depois que todos os números estiverem apagados, pressione novamente as teclas 3, +, 2 e 5. Veja que o comando ficou como você queria:

Pressione a tecla NEW LINE para ver o resultado.

SEGUNDA MANEIRA DE CORRIGIR

Às vezes, quando o comando é muito comprido, não vale a pena apagar tudo para corrigir alguma coisa bem no começo.

É possível voltar até onde estava o erro, sem ser preciso apagar.

Observe o teclado de seu TK-85.

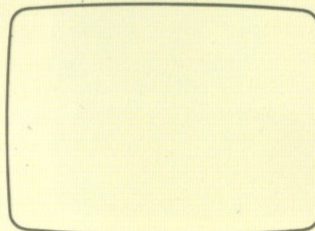
A tecla 5 tem, na parte de cima, uma flechinha virada para a esquerda.

E a tecla 8 tem uma flechinha virada para a direita.

Com essas duas teclas, podemos mover o cursor para trás e para diante, na linha de comando, para corrigir o que quisermos.

TESTE

2.1. O que devemos fazer para ligar um microcomputador?

2.2. Desenhe aqui o que aparece na tela, quando o computador é ligado:

2.3. Defina com suas próprias palavras:
CURSOR:
CARACTERES DE TEXTO:
RÓTULOS DA TECLA:
BIP:

2.4. O que significa para o computador, quando o cursor está com as letras:
[K]:

[L]:

2.5. Cite as três coisas para que serve uma tecla do microcomputador que você já conhece:

2.6. Sem usar o computador, escreva qual será o resultado dos seguintes comandos:
a. PRINT 345 − 12 + 8
b. PRINT 968 − 234 + 2

2.7. Que tecla devemos pressionar para aparecer de novo a tela com o cursor K, depois de ter sido mostrado o resultado de um comando PRINT?

2.8. Que tecla devemos pressionar quando queremos introduzir no computador algum caracter ou palavra marcado em vermelho na tecla?

2.9. Que tecla devemos pressionar para que o computador realize o cálculo dentro de um comando PRINT?

2.10. Faça um desenho mostrando como se ligam entre si as partes de um computador (vídeo e teclado):

Vamos seguir um exemplo para ver como ambas funcionam.

Suponhamos que você queira escrever este comando:

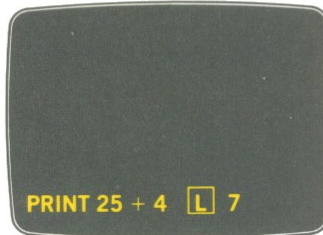
PRINT 65 + 47

E errou, escrevendo:

PRINT 25 + 47

Escreva este comando no computador:

Veja como o cursor fica no final da linha de comando. Para apagar apenas o número 2 (errado), sem apagar os outros, usamos a flechinha voltada para trás ←:

Aperte SHIFT e a tecla 5 ao mesmo tempo:

Veja como o cursor voltou a ficar entre os números 4 e 7:

Agora torne a pressionar as teclas 5 e SHIFT mais 3 vezes.

Cuidado para não esquecer de pressionar SHIFT, que fará aparecer o número 5 na linha!

Agora o cursor está exatamente depois do número que queremos corrigir. Pode pressionar a tecla RUBOUT (não se esqueça que isto se faz pressionando ao mesmo tempo as teclas SHIFT e 0):

Para introduzir o número correto, basta pressionar a tecla com o número 6. Veja como o 6 entra perfeitamente:

Se agora você quiser ver o resultado do comando, pressione a tecla NEW LINE.

A TECLA NEW LINE PODE SER PRESSIONADA COM O CURSOR EM QUALQUER POSIÇÃO.

Entendido?

Se você apertou a tecla com a flechinha ← muitas vezes, pode acontecer que o cursor vá para o começo da linha:

Para fazê-lo voltar até a posição correta, pode usar a tecla com a flechinha para a direita →, que é usada pressionando-se ao mesmo tempo as teclas SHIFT e 8:

Note que a palavra PRINT é "pulada" de uma vez só, quando se aperta uma das teclas com flechinhas.

Experimente apagar com RUBOUT a palavra PRINT. Note que ela também é apagada de uma só vez.

Introduza no computador as linhas de comando abaixo, uma de cada vez, e pressione a tecla NEW LINE para ver o resultado.

Olhe bem para a linha antes de pressionar a tecla NEW LINE.

1) PRINT 43 + 65

2) PRINT 235 − 123

3) PRINT 49 + 22 − 3

4) PRINT 894 − 425 + 11

5) PRINT 123 + 456 + 789 − 56

6) PRINT 11 + 45 − 234 − 123

7) PRINT 9 + 12 − 78

8) Introduza um comando para realizar a soma dos números 12, 24 e 33

9) Se o Zico já fez 765 gols, quantos faltam para completar 1000?

10) Introduza um comando para somar as idades de três amigos seus.

Agora introduza as linhas de comando abaixo, uma de cada vez. Antes de pressionar a tecla NEW LINE, corrija a linha de comando, usando as teclas ⇐ ⇒ e RUBOUT, de modo a ficar igual à segunda linha. Depois pressione a tecla NEW LINE e anote o resultado:

Errado	Certo	Resultado
11) PRINT 957 + 125	PRINT 957 + 124	
12) PRINT 23 − 13 + 45	PRINT 23 + 13 + 45	
13) PRINT 462 + 234 − 12	PRINT 562 − 234 − 12	
14) PRINT 123 − 1 + 85	PRINT 1213 + 12 + 85	
15) POKE 43 + 12 − 55	PRINT 43 + 12 − 55	
16) POKE 23412 + 998756	PRINT 341 + 998	
17) PRINT 1213 − 2342	PRINT 1213 + 345 − 2342	
18) PRINT 908J478	PRINT 808 + 478	
19) PRINT 1.234 − 45.5	PRINT 12.34 − 45.5	
20) PRINT P 23	PRINT 23	

Erros que o computador acusa

O computador não aceita comandos que estejam escritos de forma errada para ele. Se isso acontecer, ele mostra na tela o erro, e espera que você o corrija.

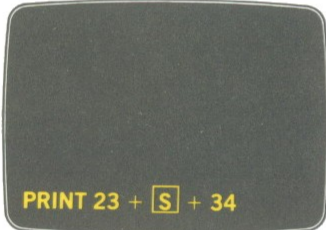
Introduza no computador a seguinte linha:

PRINT 23 + + 34

A tela fica assim:

Observe que até agora o computador ainda não notou nenhum erro.

Pressione agora a tecla NEW LINE, para que o computador realize o cálculo.

Note o que aconteceu:

Apareceu outro quadradinho, semelhante ao cursor, mas com a letra S dentro.

Mesmo sendo parecido, o S não é um cursor. Ele é uma mensagem de erro, escrita pelo computador para avisar que você escreveu algo errado no comando (no caso, dois sinais + + em seguida).

Note que o S fica exatamente depois do primeiro sinal. Isso quer dizer que o computador indica que o segundo sinal está errado (está sobrando).

O ERRO SÓ FOI NOTADO PELO COMPUTADOR AO SE PRESSIONAR NEW LINE

S indica um erro de sintaxe. Sintaxe quer dizer forma correta de escrever um comando para o computador.

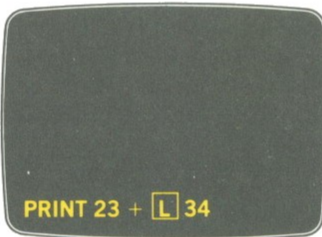
A linguagem BASIC é muito simples, mas o computador não vai entender nada que for escrito fora das regras dessa linguagem. Felizmente ele indica o erro, para que possamos corrigi-lo.

Enquanto você não corrigir esse erro, o computador não obedece ao comando.

Experimente pressionar NEW LINE de novo. Note que o computador insiste na mesma mensagem e não obedece a um comando errado.

Para corrigir o comando errado, use as teclas ← e RUBOUT, exatamente como você treinou antes.

Note que, ao pressionar a tecla ← para voltar ao local do erro, a mensagem S desaparece. O computador reconheceu que você está querendo corrigir a linha, e retirou a mensagem para não atrapalhar.

Agora apague o segundo sinal + com RUBOUT:

E pressione NEW LINE para ver o resultado:

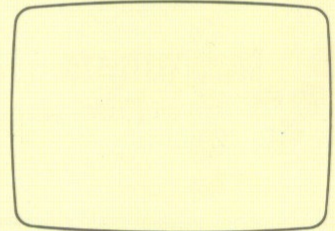
Agora o computador aceitou a linha de comando.

TESTE

3.1. Qual é a tecla usada para apagar alguma letra ou número da tela?

ATENÇÃO! Nos exercícios seguintes não use o computador!

3.2. Suponhamos que foi introduzida esta linha no computador:
PRINT 123 + 456
a. Escreva como ficará a linha se eu pressionar RUBOUT três vezes:
b. E, agora, se em seguida eu pressionar as teclas 2, 3 e 5.
c. E, agora, se eu pressionar a tecla NEW LINE:

3.3. Suponhamos que foi introduzida esta linha no computador:
PRINT 325 + 459
a. Escreva como ficará a linha, se eu pressionar a tecla ← duas vezes:
b. E, agora, se eu pressionar a tecla ← mais uma vez:
c. E, agora, se eu pressionar a tecla → por duas vezes, depois a tecla RUBOUT uma vez, e depois o número 5:
d. E como ficará o resultado, se agora eu pressionar NEW LINE:

3.4. Diga o que está errado nos seguintes comandos:
a. PRINT 345 + + 825
b. PRINT 34J902
c. PRINT PRINT 345

3.5. O que significa a mensagem S?

3.6. O que acontece com a mensagem S, quando eu pressiono RUBOUT, <= ou =>

3.7. Que tecla temos que pressionar para o computador verificar se existe algum erro na linha de comando?

EXERCÍCIOS

Introduza as linhas de comando abaixo, uma de cada vez. Note que cada uma delas tem um erro. Anote o erro que você acha que a linha tem, antes de introduzi-la no computador.

Agora pressione a tecla NEW LINE. Veja onde aparece a mensagem S. Corrija, e pressione NEW LINE de novo. Anote o resultado.

Errado	Qual é o erro?	Resultado
1) PRINT 45 + − 25	______	______
2) PRINT 23 − − 2	______	______

COMPUTADOR É UMA CALCULADORA!

Como você viu, usando a instrução PRINT, nós podemos fazer contas com o computador. Até agora, só aprendemos a fazer somas e subtrações bem simples, como

PRINT 34 + 25

Agora vamos aprender como fazer contas bem mais complicadas. Você vai ver que o seu microcomputador pode achar o resultado de sentenças ou expressões matemáticas, com todos os tipos de operações (inclusive multiplicação e divisão).

PARA USAR O COMPUTADOR COMO UMA CALCULADORA, PRECISAMOS SEMPRE COLOCAR UM COMANDO PRINT ANTES DA EXPRESSÃO MATEMÁTICA.

Esse modo de operação é chamado de MODO DIRETO.

Multiplicação e Divisão

Você já aprendeu que usamos os sinais + (mais) e − (menos), para ordenar ao computador que faça uma soma ou uma subtração.

Isso quer dizer que o computador usa os mesmos sinais que aprendemos em Matemática.

Mas, para a multiplicação e a divisão, a operação é um pouco diferente na linguagem BASIC:

A MULTIPLICAÇÃO USA O SINAL * EM VEZ DE X

A DIVISÃO USA O SINAL / EM VEZ DE ÷

Por exemplo:

Quanto é 34 multiplicado por 5?

Digite a linha PRINT 34 * 5 no teclado, e aperte a tecla NEW LINE:

Vai dar o resultado: 170

Outro exemplo:

Quanto é 14 dividido por 3?

Digite a linha PRINT 14/3 no teclado, e aperte a tecla NEW LINE:

Vai dar o resultado:

Por que o computador não aceita os sinais x e ÷, como na Matemática? Você mesmo pode imaginar por quê:

O sinal de vezes (x) pode ser confundido com a letra x

O sinal de dividir (÷) não existe no teclado.

Mas por que não posso usar o sinal : para a divisão? A resposta é igual à anterior: o computador o confundiria com dois pontos. Dois pontos e a letra x significam outras coisas para o computador, por isso não podem ser usados. Certo?

EXERCÍCIOS

Vamos fazer alguns exercícios de multiplicação e divisão no computador, usando a instrução PRINT. Ponha a resposta ao lado, e também o resultado da expressão (use o computador!):

1. 2.5 x 4.8
2. 1234 x 0.345 x 2
3. 900 x 34
4. 894 x 0.888
5. 345 : 898
6. 0.234 : 234
7. 899 : 9
8. 1 : 3
9. Uma hora tem 360 segundos. Quantos segundos tem um dia? Resposta:

10. Meu pai me dá Cr$ 30 000,00 de mesada. Quanto eu ganho por dia? Resposta:

11. Invente cinco exercícios usando divisão ou multiplicação de números:

a.

b.

c.

d.

e.

Combinando operações

A linguagem BASIC consegue fazer contas contendo várias operações matemáticas misturadas: somas, subtrações, multiplicações e divisões. É só colocar na instrução PRINT os números e operações que desejarmos, exatamente como em uma fórmula.

Por exemplo:

3 + 5 / 4

Uma combinação como esta é chamada de expressão aritmética, porque tem apenas números e operações aritméticas. Existem outros tipos de expressões, que você irá aprender depois.

Vamos resolver um probleminha, como exemplo:

Paulo tem 25 moedas de 5 cruzeiros e 14 notas de 200 cruzeiros. Qual o total de dinheiro que ele tem?

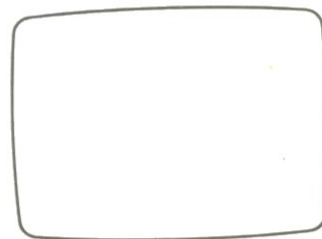
SOLUÇÃO

Como em todo problema matemático, primeiro vamos escrever a sentença que dá a solução:

total = 25 x 0.50 + 14 x 200

A instrução em BASIC para o computador fica assim:

PRINT 25 * 0.50 + 14 * 200

Coloque no computador e veja o resultado. Qual foi?

COMO O COMPUTADOR FEZ ESSA CONTA?

Observe que, para acertar nesta conta, precisamos primeiro fazer as multiplicações:

25 x 0.50 + 14 x 200
1.25 + 2800
2801,25

Ou seja, existe uma ORDEM DE REALIZAÇÃO das operações: a multiplicação é sempre realizada antes de uma soma ou subtração. Se tivéssemos somado antes, o resultado sairia errado.

Como em Matemática, a linguagem BASIC também faz contas seguindo uma certa ordem (chamada ORDEM DE PRECEDÊNCIA, porque algumas operações devem preceder outras, ou seja, serem feitas antes).
Preste bem atenção nessas regras:

1. AS DIVISÕES SÃO SEMPRE FEITAS EM PRIMEIRO LUGAR

2. AS MULTIPLICAÇÕES SÃO REALIZADAS DEPOIS DAS DIVISÕES

3. AS SOMAS E SUBTRAÇÕES SÃO FEITAS EM ÚLTIMO LUGAR

Vamos fazer algumas contas no computador, usando a instrução PRINT:

1. A altura do King-Kong foi medida (depois de morto), sendo igual a 37 pés e 8 polegadas. Quantas polegadas ele tinha de altura?

SOLUÇÃO:
Um pé (no sistema norte-americano de medidas) tem 12 polegadas
Altura = 12 x 37 + 8
PRINT 12 * 37 + 8

Coloque no computador e anote o resultado. Qual é a operação que o computador irá realizar primeiro?
Resposta:

2. O comprimento do monstro de Loch Ness é de 50 pés e 5 polegadas. Quantos centímetros ele mede?
SOLUÇÃO:

Veja agora que eu tenho que fazer uma expressão mais difícil. Primeiro eu multiplico 50 pés por 12 e somo 5, para ter o resultado em polegadas. Depois multiplico tudo por 2.54, que é o valor de uma polegada em centímetros.

Se eu fizer:
PRINT 50 * 12 + 5 * 2.54
vai dar errado, pois primeiro o computador vai realizar a multiplicação de 5 por 2.54. Para evitar isso, posso usar parênteses, como na Matemática:
PRINT (50 * 12 + 5) * 2.54

Agora vai dar certo, pois primeiro o computador faz as contas dentro dos parênteses. Depois multiplica o resultado por 2.54.

Tente fazer no computador este exercício.
Resposta:

3. O jogador Zico tinha 300 000 dólares no banco. Depois recebeu como luvas metade de 456 784 dólares, menos o imposto de 40 000 dólares. Quantos cruzeiros ele terá no total? (Imagine que 1 dólar valia, na época, 556 cruzeiros.)
SOLUÇÃO:

O total de dólares no banco será:
PRINT 300 000 + (456 784 − 40 000)/2
ATENÇÃO: Como você já deve ter notado, não se usa nas calculadoras o ponto para separar centenas, milhares etc., quando se coloca um número no computador. O ponto é usado pela linguagem BASIC como uma vírgula (para separar a parte inteira da parte fracionária de um número). Esse é o sistema usado nos Estados Unidos, onde a linguagem BASIC foi inventada.

Para obter em cruzeiros, multiplicamos tudo por 556. Para formar apenas uma expressão, temos que usar outros parênteses:
PRINT (300 000 + (456 784 − 40 000)/2) * 556

Coloque no computador e veja qual é o resultado.
Resposta:

Agora você aprendeu uma coisa importante:
A EXPRESSÃO MATEMÁTICA PODE CONTER PARÊNTESES DENTRO DE PARÊNTESES
e parênteses dentro de parênteses dentro de parênteses... quantos se quiser usar.

Está vendo como é parecido com o uso de chaves e colchetes, que você aprendeu em álgebra?

Só há uma diferença: o computador usa sempre parênteses. A linguagem BASIC não aceita colchetes ou chaves.

Por isso algumas expressões podem ficar muito confusas, e você pode esquecer de colocar algum parêntese, ou colocá-lo errado.

Lembre-se da seguinte regra:
O NÚMERO DE PARÊNTESES À DIREITA DEVE SER IGUAL AO DE PARÊNTESES À ESQUERDA.

Depois de escrever a expressão, conte o número de parênteses. Se eles não forem pares, há um erro em algum lugar.

EXERCÍCIOS

1. Escreva a instrução PRINT para as expressões matemáticas abaixo (depois coloque no computador e anote o resultado):
a. 4 787 : 2 + 455 × 855
b. 12 + 34 + 45 × 0.5
c. 34.2 + 87 : 33
d. $35.4 + \frac{75 - 42}{89 - 68}$
e. $\frac{(12.8-5.6 \times 45)}{2}$
f. $\frac{(35-7)}{2} - \frac{(45-8)}{2} \times 1.7$

2. Assinale o que está errado nas expressões abaixo. Depois coloque no computador e anote o resultado.
a. PRINT 45 * (63 − 12
b. PRINT 83 + 45) / 22
c. PRINT (23.45 + 34.2/8 − 3))
d. PRINT 34.5 * 12 − 3,44
e. PRINT (33 − [22 * 11])/2
f. 895 − 43.3 * 12

3. Um carro tinha 25.6 litros de combustível no tanque. Depois de uma viagem, ficou apenas com 17.3. Se o litro de gasolina custasse 1 688 cruzeiros, quanto custaria a gasolina gasta na viagem?
Resposta: PRINT
Resultado:

4. O Palmeiras fez 25 gols no campeonato e levou 13 gols. Cada gol a favor vale 5 pontos positivos, cada gol contra vale 7 pontos negativos. Com quantos pontos ficou o Palmeiras?
Resposta: PRINT
Resultado:

5. Quantos segundos tem um ano?
Resposta: PRINT
Resultado:

6. Qual é a média das notas 5.6, 8.2 e 7.5?
Resposta: PRINT
Resultado:

Nem sempre o seu computador vai mostrar a resposta do modo como você quer. Preste bem atenção nos exemplos seguintes e entenderá por quê.

Experimente colocar no computador a seguinte expressão:

PRINT 2/3

Veja como ficou a resposta:

Você deve saber que o resultado desta conta tem um número infinito de 6 depois do ponto:
0.66666666666666666666666... etc.

O computador mostrou apenas 6 casas decimais. Além disso, ele colocou um 7 na última casa decimal. Isto é feito sempre que a sétima casa for igual ou maior do que 5, e se chama ARREDONDAMENTO.

Suponha que o resultado de uma operação é 1.3411123443. O que o computador mostrará na tela?

Resposta:

Agora suponha que o resultado é 123.5555. O que o computador mostrará na tela?

Resposta:

O computador sempre vai mostrar um máximo de 6 algarismos. A não ser que haja menos de 6, quando mostrará apenas esses.

Por exemplo: digite a instrução:

PRINT 2/4

O resultado mostrado pelo computador na tela é:

0.5

Note que ele não mostra:

0.500000

Todos os zeros depois do 5 não são mostrados.

Funções matemáticas

Além das quatro operações aritméticas (soma, subtração, divisão e multiplicação), o computador consegue fazer outros tipos de cálculos mais complicados.

Por exemplo, se quisermos que o computador extraia uma raiz quadrada de um número qualquer, ele consegue fazer isto através da linguagem BASIC.

Para isso usamos uma instrução especial, chamada FUNÇÃO.

No nosso exemplo, a função que queremos calcular é a raiz quadrada. Em BASIC, pedimos ao computador para calcular a raiz quadrada por meio da abreviação SQR (que vem do termo SQUARE ROOT, que quer dizer raiz quadrada, em inglês).

PRINT SQR 16

Note que devo colocar as coisas em uma certa ordem, na instrução:

PRINT é o comando que diz ao computador: mostre o cálculo feito.

SQR indica ao computador que o cálculo que eu quero fazer é uma raiz quadrada. Este é o nome da função.

16 diz ao computador qual é o número do qual vai ser tirada a raiz quadrada. Ele é chamado argumento da função.

Experimente digitar essa linha no computador.

1. Para introduzir a instrução PRINT, pressione a tecla P:

2. Para introduzir a função SQR, veja que ela está indicada em cima da tecla H. Mas o cursor na tela é o L. Isto quer dizer que, se apertarmos a tecla H, aparecerá a letra H, lembra-se? Para aparecer a função SQR, devemos antes pressionar a tecla SHIFT, bem como a tecla NEW LINE.

Pressione e veja o que acontece:

O cursor mudou para F.

Isto quer dizer que, se eu apertar a tecla H, aparecerá o que está indicado logo acima da tecla (a função SQR), e não a letra H.

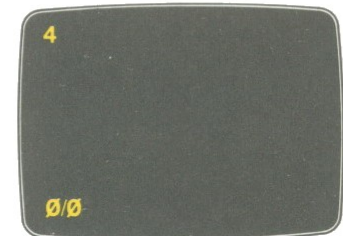
Toda vez que eu quiser introduzir no computador uma função, tenho que transformar o cursor na tela para F.

Isto é feito ordenando-se uma FUNCTION, que se consegue pressionando-se SHIFT e NEW LINE, ao mesmo tempo (note que FUNCTION está escrito em vermelho, acima da tecla NEW LINE).

Pressione agora a tecla H, e veja o que acontece:

Note que o cursor voltou a ser L novamente.

Isto quer dizer que o computador está esperando que você introduza o argumento (número):

Pressione NEW LINE e veja o resultado do cálculo:

Experimente realizar no computador os exercícios abaixo. Escreva a instrução PRINT e coloque o resultado ao lado:

a. 25
b. 10
c. 100
d. 3.14159
e. 2

Uma função matemática também pode ser colocada dentro de uma expressão aritmética:

PRINT 25 + SQR 19

O computador vai calcular primeiro a função, antes de realizar a soma. Isso quer dizer que calcular uma função vem antes, na ordem de precedência.

Coloque no computador e veja o resultado.

Outro exemplo:

PRINT SQR 25 + SQR 9

O resultado é 8, porque o computador calcula primeiro o resultado das duas funções, depois soma os resultados.

Agora, se eu fizer:

PRINT SQR (25 + 11)

O resultado será 6, certo?

O uso dos parênteses forçou o computador a fazer primeiro a conta 25 + 11 e só depois tirar a raiz quadrada.

EXERCÍCIO

Para treinar, vamos fazer o seguinte exercício:

Elenice atravessou um campo de futebol com 100 m de comprimento por 40 m de largura (veja o desenho ao lado). Quantos metros ela teve que andar?

SOLUÇÃO:
O problema é de geometria. Um triângulo como o da figura tem a seguinte expressão para calcular a hipotenusa a partir dos catetos (teorema de Pitágoras, lembra-se?):

$$A^2 = B^2 + C^2$$

Então, a instrução PRINT para resolver o problema é simplesmente:

```
PRINT SQR (B*B + C*C)
```

Veja que B*B é a mesma coisa do que B^2.
Existe também uma outra maneira de representar um expoente em BASIC: é através do sinal ** (dois asteriscos). Ele está em cima da letra H, e, para conseguir colocá-lo na tela, basta apertar a tecla SHIFT juntamente com a tecla H.

Tente:

```
PRINT SQR (B**2 + C**2)
```

Cuidado para não apertar a tecla * duas vezes: na tela vai aparecer a mesma coisa, mas o computador não vai aceitar. Tem que apertar a tecla que dá dois asteriscos de uma vez. OK?

Escreva aqui embaixo o resultado:

Resposta:
A exponenciação (um número elevado a outro) é uma operação com ordem de precedência mais alta do que a divisão. Ou seja:

SE HOUVER UMA EXPONENCIAÇÃO, ELA SERÁ CALCULADA EM PRIMEIRO LUGAR

Por exemplo:

```
PRINT 2 + 3**2/3
```

Dará o resultado 5, porque o computador primeiro calculará 3**2 (que é 9), depois dividirá o 9 por 3 (pois a divisão é a que vem depois). Aí ele fará a soma.
Existem muitas outras funções no BASIC, além da raiz quadrada. Depois aprenderemos outras.

TESTE

4.1. Escreva os símbolos usados em BASIC para as seguintes operações aritméticas:
a. adição
b. subtração
c. multiplicação
d. divisão

4.2. Por que não são usados os símbolos × e : para multiplicação e divisão em BASIC?
Resposta:

4.3. Numere as operações aritméticas, de acordo com a ordem em que são efetuadas, dentro de uma expressão em BASIC:

Subtração
Multiplicação
Divisão
Adição

4.4. Coloque as expressões abaixo em forma de PRINT:
a. $3 \times 2 + 2$
b. $3 \times (2 + 2)$
c. $4 \times 5 - 9$
d. $3 : 2$
e. $(5 + 4) \times (5 : 9) - 1$
f. $(5 : 9) \times (3 + 2) - (1 : 2)$

4.5. Faça os seguintes exercícios utilizando o comando PRINT. Primeiro escreva a linha de instrução formulando o problema, depois coloque no computador e anote o resultado.

a. Qual é a área de um triângulo de base 120 e altura 25? (Dica: ÁREA = BASE × ALTURA.)
Resposta:

b. Qual é a soma das áreas de um triângulo de base 32 e altura 44 e um quadrado de lado 24? (Utilize apenas uma instrução PRINT.)
Resposta:

c. Qual é o volume de um cubo com 32,8 cm de aresta?
Resposta:

d. Um campo de futebol americano mede 100 jardas. Cada jarda mede 36 polegadas. Qual é o comprimento do campo em centímetros e em metros? (Dica: uma polegada vale 2,54 cm.)
Resposta:

e. Uma motocicleta gastou 25,8 litros de gasolina em uma viagem. No começo da viagem o mostrador de quilometragem indicava 12346 e, ao final da viagem, indicava 13193. De quantos quilômetros por litro foi o consumo?
Resposta:

f. A circunferência de um círculo é dada pela fórmula $2 \times \pi \times R$. (R é o raio.) Uma bicicleta tem a roda com o raio de 35,234 cm. Qual é a circunferência da roda? (Dica: a constante π vale 3,14159.)
Resposta:

g. Quantas voltas por quilômetro dará uma roda de automóvel com um diâmetro de 45 cm?
Resposta:

h. Em um triângulo retângulo, o quadrado da hipotenusa é igual à soma dos quadrados dos catetos. Supondo que a hipotenusa mede 29,238 e um dos catetos mede 12,34, qual é o valor do outro cateto?
Resposta:

4.6. Verifique se os comandos abaixo estão corretos. Se não estiverem, coloque ao lado a forma correta. Depois, digite no computador e escreva o resultado ao lado.

a. PRINT (2 * (3 + 4) − 2

b. PRINT PRINT

c. PRINT 84 * 1135−20*4767

d. PRINT − 3 * − 4

4.7. Para que servem os parênteses em uma expressão matemática em BASIC?
Resposta:

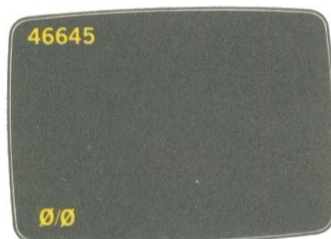
4.8. O que substitui as chaves e colchetes de uma expressão matemática em BASIC?
Resposta:

VAMOS PROGRAMAR O COMPUTADOR?

Quando quisermos que o computador faça algum cálculo e mostre o resultado na tela, usamos o comando PRINT. Este comando, junto com a expressão matemática, funciona como uma ordem para o computador.

Assim, se você escrever no teclado esta instrução:

PRINT 34534 + 12111

e pressionar a tecla NEW LINE, o computador fará a conta e mostrará o resultado na tela:

E SE VOCÊ QUISER FAZER DE NOVO A MESMA CONTA NO COMPUTADOR?

Você já sabe a resposta:

A instrução precisa ser introduzida de novo no computador.

Se você tiver que fazer uma série de contas repetidas, isso vai dar muito trabalho, porque cada vez você vai ter que digitar de novo as instruções.

Felizmente o computador tem MEMÓRIA.

E existe um macete para colocar na memória dele as instruções que quisermos. Depois de termos feito isso, ele vai se lembrar das instruções, e não vai ser preciso digitá-las de novo toda vez que quisermos repetir os cálculos.

Preste bem atenção:

PROGRAMAR O COMPUTADOR É COLOCAR INSTRUÇÕES EM SUA MEMÓRIA.

Você vai aprender a fazer um PROGRAMA.

Um Programa em BASIC

No capítulo 3, vimos que o computador pode ser usado como uma calculadora, através do MODO DIRETO.

O computador obedece a instrução PRINT que damos a ele, e em seguida "esquece" a instrução.

Fazer com que o computador grave a instrução em sua memória é a coisa mais simples do mundo: BASTA COLOCAR UM NÚMERO QUALQUER NO COMEÇO DA INSTRUÇÃO.

Siga o exemplo:

Se quisermos colocar na memória a seguinte instrução:

PRINT 34534 + 12111

basta digitar (escrever no teclado) da seguinte forma:

1 PRINT 34534 + 12111

Vamos entender juntos tudo o que está contido nessa linha.

1 — é o número da linha. Pode ser qualquer número;
PRINT — é a instrução, ou comando que damos ao computador;
34534 — é um dos dados que fazem parte da expressão matemática de cálculo;
+ — é a operação que queremos fazer (que, no exemplo, é a de somar);
12111 — é o outro dado da expressão de cálculo.

Experimente digitar essa linha no computador, e apertar a tecla NEW LINE:

Veja o que aconteceu:

O COMPUTADOR NÃO FEZ A CONTA!

Ao invés disso, ele colocou a linha que você introduziu lá no topo da tela.

ISSO QUER DIZER QUE ELE GUARDOU A INSTRUÇÃO NA MEMÓRIA!

O número 1 significa que essa é a primeira instrução que queremos que ele obedeça. Depois você vai ver que podemos colocar mais instruções EM UM MESMO PROGRAMA.

Note também que a instrução que estava digitada na linha de baixo da tela desapareceu, como se fosse carregada para a parte de cima.

E apareceu de novo o cursor K. Isso quer dizer que o computador está pronto para aceitar mais uma ordem.

E agora?

Como vamos fazer para obrigar o computador a obedecer a instrução PRINT?

Ele guardou na memória, tudo bem. Mas, para obrigá-lo a EXECUTAR O PROGRAMA (ou seja, obedecer as instruções), eu devo usar um novo comando:

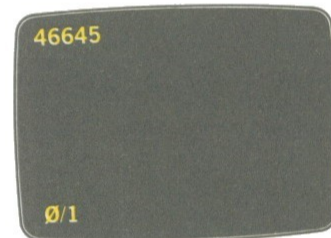
RUN

e pressionar a tecla NEW LINE.

RUN, em inglês, quer dizer CORRA! Que significa também: RODE O PROGRAMA. Rodar, na gíria das pessoas que lidam com computador, quer dizer: executar um programa que estiver na memória do computador.

Para introduzir o comando RUN, aperta-se a tecla R, quando o cursor na tela for K. Note que acima da tecla R está escrito RUN.

Pressione RUN e veja o que acontece:

E agora aperte a tecla NEW LINE:

o resultado no cálculo apareceu na tela, em cima.

Na linha de baixo, a mensagem 0/1 significa:

0 — não houve nenhum erro no programa
1 — o programa parou na linha número 1

Agora que a instrução está na memória do computador (no programa), podemos pedir para o computador executá-la novamente quantas vezes quisermos. Basta digitar RUN novamente. Experimente!

Qual é o resultado?

Pressione agora a tecla NEW LINE de novo. Veja o que acontece:

O programa apareceu de novo no topo da tela.

Isto quer dizer que ele ainda está na memória. Se ele está lá ainda, quer dizer que eu posso executá-lo novamente com outro comando RUN.

O cursor K também voltou a aparecer na parte de baixo da tela.

Fazendo Programas

Outra vantagem de colocar as instruções em um programa é poder obrigar o computador a memorizar mais de uma instrução.

Da última vez que você mexeu no computador, a tela estava assim:

Agora vamos colocar uma outra instrução no programa. Para isso bastará colocar outro número no começo do novo programa. Esse número deverá estar em ordem crescente, comparado com o número do programa anteriormente digitado. Como a instrução anterior usou o número 1, vamos usar nesta o número 2:

2 PRINT 897 – 345

Digite essa nova instrução e pressione a tecla NEW LINE:

Note que agora o nosso programa tem duas instruções, uma debaixo da outra. Por isso demos o número 2 (ordem crescente) à segunda linha. É para que o computador saiba que essa linha deve vir depois da outra: assim ele vai saber a ordem das linhas.

Pressione RUN e NEW LINE e veja o resultado:

o computador calculou e mostrou primeiro o resultado da linha 1 e depois o resultado da linha 2.

Ele sempre obedece assim aos programas: primeiro ele obedece a um comando e só depois vai obedecer ao outro.

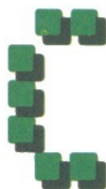
Se você pressionar NEW LINE novamente, veja o que acontecerá:

Aí está o programa novamente.

Outra coisa interessante de um programa BASIC é que eu posso mandar o computador rodá-lo (executá-lo) a partir da linha que eu quiser. Experimente digitar RUN 2:

Veja o resultado:

O computador fez apenas a segunda conta, ou seja, ele executou o programa a partir da linha número 2.

SE EU NÃO COLOCAR NENHUM NÚMERO APÓS O COMANDO RUN, ele começa a executar um programa a partir da instrução que estiver em primeiro lugar.

Como Apagar um Programa

Se eu quiser tirar este programa da memória do computador, para colocar outro, existe o comando:

NEW

NEW quer dizer NOVO, em inglês. Ou seja, eu quero colocar um programa novo, por isso apago da memória o programa velho.

Para introduzir o comando NEW, deve-se apertar a tecla A, quando na tela estiver o cursor K.

Experimente:

E agora aperte a tecla NEW LINE. Note que o programa desapareceu da memória.

EXERCÍCIO

Vamos fazer um programa para entender melhor o que foi explicado. O problema vai ser o seguinte:

Vamos fazer um programa para mostrar na tela a tabuada de multiplicar pelo número 7.

O programa para fazer isso deve ser assim:

```
10 PRINT 7 * 1
20 PRINT 7 * 2
30 PRINT 7 * 3
40 PRINT 7 * 4
```

Coloque esse programa na memória do computador.

Você deve estar achando estranho que os números das linhas neste programa vão de 10 em 10.

Isso não é problema. Como já foi dito, a gente pode colocar o número de linhas que quiser, desde que esteja em ordem crescente (ou seja, os números das linhas que vêm depois têm que ser cada vez maiores). Existe uma vantagem em colocar os números de linha de 10 em 10 ou de 20 em 20, em vez de 1 em 1, como no primeiro programa. Depois você vai entender por quê.

Por enquanto, introduza o programa como está aqui.

E pressione a tecla RUN. O resultado mostrado na tela será:

Apareceu tudo de uma vez, uma multiplicação em cada linha.

Corrigindo o Programa

O programa que está colocado na memória pode ser modificado quantas vezes eu quiser.

Existem maneiras de colocar mais linhas, digitar de novo uma linha (se eu errei ou quero mudá-la), e até apagar uma linha.

Essas são chamadas funções de edição do BASIC.

Observe que eu posso colocar mais linhas NO FINAL do programa, se eu quiser:

50 PRINT 7 * 5

Pressione RUN.

O que aconteceu?

Se uma linha do programa na memória estiver errada, ou se eu quiser mudá-la, basta escrevê-la de novo, começando com o mesmo número da linha:

Vamos supor que eu quero a linha 50 mudada para:

50 PRINT 7 * 6

Experimente digitá-la assim, e pressione NEW LINE. Veja o que acontece:

A linha 50 foi SUBSTITUÍDA por uma nova. A linha 50 antiga sumiu.

Posso também colocar novas instruções NO MEIO do programa que está na memória.

Suponhamos que eu quero ver também o resultado de 7 vezes 2,5 (aparecendo na tela entre 7 x 2 e 7 x 3). Experimente digitar a linha.

25 PRINT 7 * 2.5

E pressione a tecla NEW LINE:

Agora pressione a tecla RUN:

O COMPUTADOR COLOCA A LINHA DE INSTRUÇÃO NA ORDEM CERTA

Agora você entendeu qual é a vantagem de numerar as linhas de 10 em 10. Deixando um "espaço" entre elas, eu posso mais tarde colocar uma ou mais linhas de instrução entre elas.

Isso se chama INSERIR LINHAS.

Outra coisa interessante que posso fazer é apagar uma linha sem precisar apagar todo o programa.

Com o programa da tabuada na memória, experimente digitar apenas o número de linha 25 e pressione NEW LINE:

Pressionando de novo NEW LINE, você vê que a linha 25 desaparece.

Pressione RUN e me diga qual foi o resultado:

Resposta:

T E S T E

5.1. O que se deve fazer para colocar uma instrução na memória do computador?

5.2. O que acontece quando nós digitamos uma linha de instrução que começa com um número?

5.3. O que acontece quando nós digitamos no computador uma linha de instrução que não começa com um número?

5.4. Defina com suas próprias palavras o que é um PROGRAMA:

5.5. Cite duas vantagens de poder colocar um programa na memória do computador:

5.6. Para que servem os comandos:

RUN
NEW
PRINT

5.7. Por que devemos numerar em ordem crescente as linhas de um programa?

5.8. Faça de conta que você é o computador, e alguém colocou na sua memória o programa abaixo:

10 PRINT 2 * 3/2

20 PRINT (233 – 33)/10

30 PRINT SQR 16

Agora responda às perguntas abaixo, sobre este programa.

Mas lembre-se: não vale testar no computador. Faça de conta que você é o computador!

Como apareceria o resultado na tela, se o computador receber o comando RUN?

Como apareceria o resultado na tela, se o computador receber o comando RUN 20?

Como ficaria o programa na memória, se for introduzida a linha:

15 PRINT 99

E como ficaria o programa, se agora fo digitado apenas o número:

30 (NEW LINE)

Qual será o novo resultado, se o compu dor receber o comando RUN novamente?

AGORA COLOQUE ESSES EXERCÍCIOS COMPUTADOR E VEJA SE CONFERE.

5.9. Faça um programa para calcular e m trar na tela a tabuada de multiplicar do mero 9:

5.10. Faça um programa para calcular qu tos segundos tem um ano, e quantos dias um período de 14 anos.

Programas Que Fazem Contas

Vamos fazer agora alguns exercícios utilizando o comando PRINT dentro de programas.

Para isso, vamos programar alguns exemplos do TESTE do capítulo 3:

1. Um campo de futebol americano mede 100 jardas. Cada jarda mede 36 polegadas. Qual é o comprimento do campo em centímetros e em metros (dica: uma polegada vale 2,54 cm)

SOLUÇÃO:

```
10 PRINT 100 * 36 * 2.54

20 PRINT (100 * 36 * 2.54)/100

RUN
```

A linha 10 vai dar a solução em centímetros, e a linha 20 em metros (100 jardas vezes 36 polegadas vezes 2,54 cm dividido por 100, para dar a solução em metros).

Resposta: ....

2. Uma motocicleta gastou 25.8 litros de gasolina em uma viagem. No começo da viagem o contador de quilometragem indicava 12346 e, ao final da viagem, indicava 13193. De quantos quilômetros/litro foi o consumo?

SOLUÇÃO

```
10 PRINT (13193 - 12346)/25.8

RUN
```

Resposta: ....

3. A circunferência de um círculo é dada pela fórmula $2 \times \pi \times R$. O símbolo π vale 3.14159; e o R é o raio da circunferência. Uma roda de bicicleta tem o raio de 35,234 cm. Qual é a circunferência da roda?

SOLUÇÃO:

```
10 PRINT 2 * 3.14159 * 35.234

RUN
```

Resposta: ....

4. Quantas voltas por quilômetro dará uma roda de automóvel com diâmetro de 45 cm?

SOLUÇÃO

```
10 PRINT 100 * 1000/(3.14159 * 45)

RUN
```

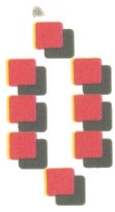
Resposta: ....

O número de centímetros contidos em um quilômetro é 100 x 1000.
A circunferência de um círculo é igual a $\pi \times$ diâmetro.
Dividindo o número de centímetros em um quilômetro pela circunferência, teremos o número de voltas dadas em um quilômetro.

5. Em um triângulo retângulo, o quadrado da hipotenusa é igual à soma dos quadrados dos catetos. Supondo que a hipotenusa mede 29,238 e um dos catetos mede 12,34, qual é o valor do outro cateto?

SOLUÇÃO:

```
10 PRINT SQR (29.238     2 - 12.34
** 2)

RUN
```

Resposta: ....

Até agora, você viu que podemos usar a instrução PRINT para mostrar números na tela do computador, de modo direto ou em um programa. O comando PRINT também serve para calcular expressões matemáticas e mostrar, ao mesmo tempo, o resultado.

Agora vamos aprender como programar o computador para que ele escreva letras na tela.

Se conseguirmos fazer com que o computador escreva letras, também poderemos conseguir que escreva nomes, títulos e mensagens, certo?

Uma instrução PRINT também pode ser usada para escrever letras (A, B, C, D... etc.), e outros sinais especiais (& % $ \ e outros). As letras, números e sinais especiais são chamados de CARACTERES.

A letra A é um caractere.
O cifrão ($) também é um caractere.

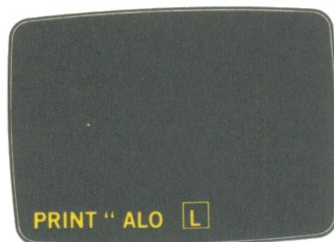
E o espaço em branco também é um caractere (tecla SPC, que quer dizer ESPAÇO, e tem o mesmo efeito que aquela barra da parte de baixo do teclado da máquina de escrever).

É muito simples fazer o computador escrever alguma coisa.

Basta colocar entre aspas a MENSAGEM que nós queremos que ele escreva.

Assim:

```
PRINT "ALÔ, EU SOU O
COMPUTADORR"
```

Experimente digitar esta linha no computador:

E aperte NEW LINE:
Viu?

O computador escreve na tela exatamente o que você escreveu entre aspas.

Mesmo se você tivesse colocado algo errado (ou um palavrão!):

```
PRINT "ALÔ, EU SOU O
COMPUTADOR"
```

Ao computador não interessa o que você põe entre aspas.

Pode ser até em turco ou em italiano. Ele reproduz como está.

```
PRINT "HELLO, HOW ARE YOU?"
```

Podemos colocar mensagens em uma instrução PRINT, tanto de modo direto (linha sem número no começo), quanto em um programa (linha com um número no começo).

Vamos fazer um programinha.

Primeiro pressione a tecla NEW para "limpar" a memória.

Introduza agora o seguinte programa:

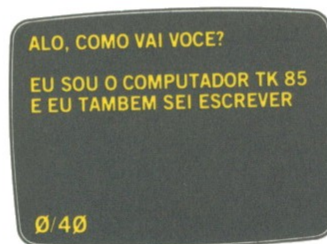
```
10 PRINT "ALÔ, COMO VAI VOCÊ?"
20 PRINT
30 PRINT "EU SOU O
   COMPUTADOR TK 85"
40 PRINT "E EU TAMBÉM SEI
   ESCREVER"
```

Você já aprendeu que o computador é meio "burro".

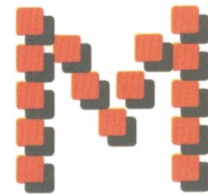
Ele só faz uma coisa de cada vez.

Primeiro, ele obedece o comando número 10, depois o 20, depois o 30.

Veja agora o que acontece, se você pressionar a tecla RUN:

```
ALO, COMO VAI VOCE?

EU SOU O COMPUTADOR TK 85
E EU TAMBEM SEI ESCREVER


Ø/4Ø
```

Usando espaços (como o da linha 20, que fica "em branco" e como os da tecla SPC, que separa as palavras entre si), eu posso colocar onde quiser, na tela, alguma mensagem ou título:

```
10 PRINT "COMPUTADOR TK 85"
```

Todos os espaços colocados na mensagem entre aspas são mostrados na tela do computador.

Usar palavras entre aspas dentro de uma instrução PRINT é muito útil para dar um título às respostas dos cálculos feitos pelo computador.

A instrução PRINT permite a você misturar mensagens com expressões, na ordem que quiser.

Vamos dar um exemplo:

Quantos segundos existem em um ano?

Esse programa você já fez:
10 PRINT 60 * 60 * 24 * 265

O computador vai mostrar este resultado, quando você pressionar a tecla RUN:

Você sabe o que significa esse número sozinho lá na tela, certo? Afinal, foi você quem resolveu o problema.

EXERCÍCIOS

1. Faça um programa para escrever na tela o seu nome e o de sua escola.
Resposta:

2. Faça um programa para colocar seu nome bem no meio da tela, na linha de cima.
Resposta:

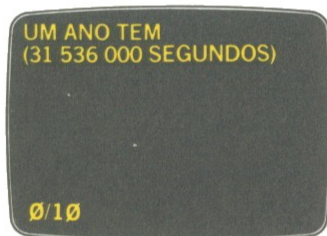
Mas os programas de computador são feitos também para uso de outras pessoas. Por isso é bom colocar títulos para explicar os resultados na tela.

Veja como fica bem melhor assim o programa:

10 PRINT "UM ANO TEM"; 60 * 60 * 24 * 365; "SEGUNDOS."

Vamos analisar as partes desta instrução:

10 ⇒ é o número da linha
PRINT ⇒ a instrução para mostrar o que vem depois
"UM ANO TEM" ⇒ deve aparecer primeiro esta mensagem
; ⇒ serve para separar a mensagem do que vem depois
60 * 60 * 24 * 365 ⇒ cálculo do número de segundos em um ano
; ⇒ serve para separar o número resultante da palavra
"SEGUNDOS." ⇒ outra mensagem

Experimente digitar o programa no computador. Pressione a tecla RUN e veja o que acontece:

Viu como funciona direitinho?

EM UMA INSTRUÇÃO PRINT NÓS PODEMOS COLOCAR QUANTOS TÍTULOS, EXPRESSÕES E NÚMEROS QUISERMOS.

A única condição é que eles estejam separados uns dos outros por meio de espaços.

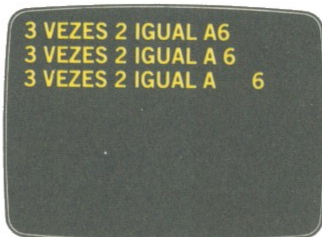
Você aprendeu que o ponto e vírgula (;) é usado para separar os elementos (partes) de uma linha de impressão PRINT.

USANDO O ; NO LUGAR DE SPC, AS PARTES DA LINHA DE IMPRESSÃO VÃO ESTAR UNIDAS UMAS NAS OUTRAS.

Ou seja, elas serão mostradas na tela sem ter entre si um espaço em branco.

Experimente que diferença faz estes três comandos PRINT:

```
10 PRINT "3 VEZES 2 IGUAL A"; 6
20 PRINT "3 VEZES 2 IGUAL A "; 6
30 PRINT "3 VEZES 2 IGUAL A    "; 6
```

Digite este programa no computador e veja o resultado quando pressionar a tecla RUN:

Quando pomos várias mensagens em seguida na mesma instrução PRINT, podemos fazer com que elas apareçam grudadas umas nas outras ou não. Tente digitar as seguintes linhas de modo direto:

```
PRINT "RENATO"; "REJANE"
PRINT "RENATO "; "REJANE"
PRINT "RENATO"; " REJANE"
PRINT "RENATO"  ; "  REJANE"
```

Responda: qual a diferença entre os resultados impressos destas linhas?

....

....

OS ESPAÇOS EM BRANCO DENTRO DAS DUAS ASPAS TAMBÉM SÃO IMPRESSOS.

Imprimindo Onde Eu Quero

Depois de saber para que serve o ponto e vírgula no comando PRINT, é fácil entender como podemos imprimir qualquer mensagem ou resultado em um ponto qualquer da tela.

Vamos mostrar um exemplo:

Eu quero imprimir na tela uma linha contendo o nome de um time de futebol (no começo da linha), depois a cidade de onde ele é (começando na posição número 15 da linha), e quantos pontos ele tem (começando na posição número 25 da linha).

Pondo vários times assim, vai se formar uma tabela bonitinha, com todas as colunas alinhadas, certo?

```
PALMEIRAS      SÃO PAULO     12
GUARANI        CAMPINAS       9
PONTE PRETA    CAMPINAS       7
```

É claro que nós podemos fazer linhas PRINT assim:

```
10 PRINT "PALMEIRAS SÃO PAULO 12"
20 PRINT "GUARANI CAMPINAS 9"
30 PRINT "PONTE PRETA CAMPINAS 7"
```

Mas é claro que assim há muito desperdício de espaços em branco.

Cada letra, número ou espaço que colocamos dentro de uma mensagem PRINT ocupa uma unidade a mais de memória no computador.

Como o computador não tem muita memória, às vezes é importante economizá-la.

Outra coisa aborrecida neste método é que temos que digitar um monte de brancos repetidos. Isso demora e sempre apresenta erros (contamos o número errado de brancos).

PARA CONTAR ESPAÇOS EM BRANCO ENTRE DUAS MENSAGENS OU NÚMEROS, EXISTE UMA FUNÇÃO ESPECIAL EM BASIC

Lembra-se das funções?

O exemplo que nós demos foi da raiz quadrada (SQR): você fornece um número (o argumento), e a função calcula outro (a raiz quadrada).

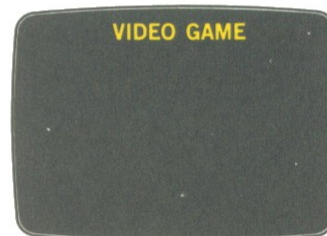
Agora vamos estudar a função TAB.

(Procure onde ela fica no teclado. Lembre-se de que é preciso transformar antes o cursor em F, pressionando SHIFT E NEW LINE).

Ela não calcula nada: apenas coloca onde quisermos os elementos de uma instrução PRINT.

Por exemplo:

PRINT TAB 10;"VIDEO GAME"

Vai escrever a mensagem:

começando na posição 10 da linha, quando o comando PRINT for obedecido.

Dizemos que houve uma TABULAÇÃO até a coluna (ou posição) 10.

Veja como ficaria melhor o nosso programa de exemplo:

```
10 PRINT "PALMEIRAS"; TAB 15; "SÃO PAULO"; TAB 25;12
" 20 PRINT "GUARANI"; TAB 15; "CAMPINAS"; TAB 25;7
30 PRINT"PONTE PRETA"; TAB 15; "CAMPINAS"; TAB 25;5
```

Digite o programa no computador (nunca se esqueça de pressionar a tecla NEW LINE se antes já tiver algum programa na memória, que possa apagá-lo).

Veja como o resultado é idêntico ao programa anterior.

E você aprendeu algumas coisinhas agora:

1. A FUNÇÃO TAB NÃO CALCULA NADA (NÃO É MATEMÁTICA)

2. O ARGUMENTO DE TAB INDICA EM QUE COLUNA VAI COMEÇAR A MENSAGEM

3. POSSO COLOCAR VÁRIOS TAB NUMA MESMA LINHA

4. A FUNÇÃO TAB FUNCIONA SOMENTE PARA UMA LINHA NA TELA

Uma outra coisinha: o argumento da função TAB também pode ser uma expressão matemática. Por exemplo:

PRINT TAB (45 - 3*2);"MEU NOME"

que é calculado pelo computador e vale a mesma coisa que:

PRINT TAB 39;"MEU NOME"

Note que temos que usar parênteses quando a expressão é constituída por letras.

Agora, para não usar de modo errado o comando TAB, é importante saber o seguinte:

UMA LINHA NA TELA DO TK 85 TEM ATÉ 32 COLUNAS

Por isso, não é bom colocar um número maior que 32 como argumento de TAB:

Não funciona colocar vários TAB numa mesma linha, mas com argumentos que sejam menores do que os que vinham antes.

Por exemplo:

PRINT TAB 12;"JOÃO E"; TAB 5;"MARIA"

Experimente no computador para ver o que acontece.

Resposta:

Existe também uma outra maneira mais simples de separar as partes misturadas de um comando PRINT, por meio de tabulações: é usando a vírgula.

A diferença entre o ponto e vírgula e a vírgula é que esta tabula, na hora da impressão, a mensagem ou resultado, começando na coluna 16 na tela.

Vamos seguir estes exemplos:

PRINT "RENATO","REJANE"

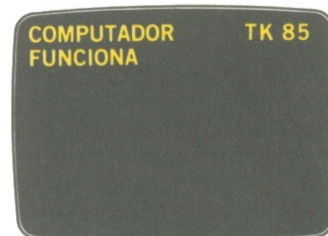
Vai aparecer assim na tela:

Note que RENATO está começando na coluna 1 da primeira linha, e REJANE na coluna 16 da mesma linha.

A vírgula funciona como TAB 16.

Agora outro exemplo:

PRINT "COMPUTADOR",TK 85", "FUNCIONA"

Vai aparecer assim na tela:

O que aconteceu?

A primeira vírgula colocou TK 85 na coluna 16 da primeira linha.

A segunda vírgula iria colocar mais 16 espaços de tabulação (o que daria 32).

Mas como a linha só tem 32 colunas, a palavra FUNCIONA foi para a primeira posição da linha seguinte!

Podemos usar a vírgula para brincar um pouco:

PRINT"FLIPERAMA", LEGAL",, TK 85"

Veja que duas vírgulas sem nada entre si, valem dois TAB 16.

EXERCÍCIOS

Vamos fazer agora alguns exercícios simples.

Vamos pegar alguns exemplos de EXERCÍCIOS anteriores de cálculo, usando o PRINT, e vamos colocar alguns títulos para melhorar a maneira como o computador mostra os resultados na tela.

Para cada programa seguinte, procure fazer várias linhas: umas contendo os dados do problema, e outras os resultados.

1. Um carro tinha 45.5 litros de combustível no tanque. Depois de uma viagem ficou apenas com 10.0. Eu tinha gasto Cr$ 12 400,00 para encher o tanque de 50 litros. Quanto custou o litro de gasolina e quanto custou a gasolina gasta na viagem?

Resposta:

Resultado:

2. Qual é a média das notas 4.5, 8.6, 3.2 e 7.0?

Resposta:

Resultado:

3. Um campo de futebol americano mede 100 jardas. Cada jarda mede 36 polegadas. Qual é o comprimento do campo em centímetros e em metros (dica: uma polegada vale 2,54 cm)?

Resposta:

Resultado:

TESTE

6.1. Digite no computador:
a. PRINT 8 * 6
b. PRINT"8 * 6"
c. PRINT"8 * 6 =";8 * 6
Qual foi o resultado em cada caso?
Explique a diferença:

6.2. Para que servem a vírgula e o ponto e vírgula dentro de uma declaração PRINT?

6.3. Qual é a utilidade de se misturar mensagens entre aspas e resultados numéricos ou expressões matemáticas em um comando PRINT?

6.4. Quais são as dimensões da tela do TK 85:
número de linhas:
número de colunas:
Para que serve a última linha de baixo na tela?

6.5. Verifique se os comandos abaixo estão corretos. Se não estiverem, coloque ao lado a forma correta. Depois, digite no computador e escreva o resultado ao lado.

a. PRINT (2 * (3+4)−2

b. PRINT PRINT

c. PRINT 84 * 1135−20 * 4767

d. PRINT "TK";85;"C"

e. PRINT "CURSO DE COMPUTADORES"

f. PRINT −3 * −4

g. PRINT"CONTAS DE SOMAR

h. PRINT"COMPITADOR

6.6. Usando a função TAB, escreva uma instrução PRINT para colocar a palavra COMPUTADOR exatamente no centro da linha da tela.

6.7. Escreva uma instrução PRINT para colocar na tela os números de 1 a 8, distribuídos em duas colunas, assim:

```
1          2
3          4
5          6
7          8
```

Resposta:

6.8. Responda: TAB é:
a. comando
b. função
c. programa
d. não existe em BASIC

6.9. Para que serve a função TAB?

6.10. Faça de conta que você é o computador. Como ficará a tela do computador, depois de obedecer ele as instruções seguintes?

a. PRINT"MEU NOME É","DANIEL"

b. PRINT TAB 25;"TESTE",,,"ESPACIAL"

c. PRINT TAB 12;12;TAB 28;28

6.11. Responda se é verdadeiro (V) ou falso (F):
( ) A função TAB não calcula nenhuma função matemática.
( ) Só se pode colocar um TAB em cada instrução PRINT.
( ) A vírgula usada para separação em PRINT funciona como TAB.
( ) TAB quer dizer tabulação.
( ) Posso usar a função TAB com um número maior do que 32.

6.12. Faça um programa para colocar na tela uma lista de 3 times de futebol, numerada. Por exemplo:
1 PALMEIRAS
2 BOTAFOGO
3 FLAMENGO
Resposta:

6.13. Faça um programa para colocar um sinal + em cada canto da tela (dica: use o comando PRINT AT).

Repetindo Coisas

Outro dia eu me convenci de que devia escrever quatro vezes meu nome na tela do computador.

O programa para fazer isso é fácil, não acha?

```
10 PRINT "RENATO"
20 PRINT "RENATO"
30 PRINT "RENATO"
40 PRINT "RENATO"
```

O computador vai obedecer assim:

Faça a mesma coisa com o seu nome e veja o resultado!

Eu também poderia ter escrito assim:

10 PRINT "RENATO RENATO RENATO RENATO"

Em vez de aparecer tudo na vertical (um nome em cada linha), ia aparecer tudo na horizontal, certo?

Pois bem. E se eu quiser encher a tela inteira com o meu nome?

Não responda já: preste bem atenção.

A tela tem 21 linhas. Eu teria que colocar no programa 21 instruções PRINT, todas iguais.

Daria muito trabalho, não é?

EU POSSO ORDENAR AO COMPUTADOR QUE REPITA UMA LINHA DE PROGRAMA QUANTAS VEZES EU QUISER.

Basta mandar que ele "volte" até a linha PRINT.

Assim:

```
10 PRINT "RENATO"
20 GOTO 10
```

A instrução GOTO, em inglês, quer dizer: VÁ PARA

Portanto, GOTO 10 quer dizer: VÁ PARA A LINHA NÚMERO 10!

Como é que isto faz o computador repetir a linha 10?

Lembra-se de que eu disse que o computador obedece (executa) uma linha de cada vez? E que ele só executa a linha seguinte depois de fazer o que a linha anterior diz?

Pois bem. Siga as flechinhas abaixo. Você é o computador.

```
10 PRINT "RENATO"
```

```
20 GOTO 10
```

À primeira vez que eu pressionar a tecla RUN, o que acontece?

O computador obedece a instrução na linha 10, e escreve na tela:

Em seguida ele vai obedecer a linha 20, que diz a ele:

VÁ PARA A LINHA 10

E ele, bonzinho, volta para lá e executa de novo a linha 10:

E de novo:

Até preencher a tela toda!

Quando ele preenche a tela, aparece na última linha a mensagem:

5/10

Isso quer dizer que o computador parou na linha 10, acusando uma mensagem número 5.

A mensagem número 5 significa:

NÃO HÁ MAIS ESPAÇO NA TELA

(veja a lista de mensagens de erros no fim da obra, onde está explicado o que significa cada uma delas).

Agora o computador vai ficar esperando que você decida o que fazer:

a. Se você quiser ficar lendo ou admirando a tela escrita, ele espera.

b. Se você quiser que o programa continue a ser executado, procure onde fica o comando CONT. Ele significa CONTINUE: você está dizendo a ele para continuar.

O que acontece?

A tela fica escura, e depois o programa continua a repetir a linha 10.

c. Quando você estiver cansado de ver o programa continuar, ao aparecer a mensagem 5/10 pressione a tecla NEW LINE:

aparece de novo o programa que estava na memória.

Uma outra maneira de interromper o programa (parar de executar), enquanto ele estiver ainda escrevendo as linhas, é pressionar a tecla BREAK.

BREAK em inglês quer dizer QUEBRAR ou INTERROMPER.

Veja onde fica a palavra, acima da tecla NEW LINE.

Por isso você precisa pressionar ao mesmo tempo a tecla SHIFT e a tecla NEW LINE.

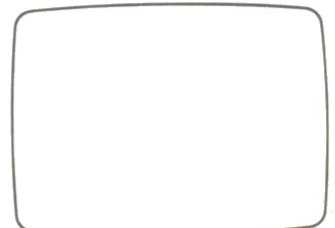
Esta tecla BREAK só funciona enquanto o programa ainda não parou, ao chegar ao final da tela.

Faça agora um programa para escrever seu nome em toda a tela (um nome em cada linha):

Como fazer para escrever o nome em toda a tela, mas um depois do outro (vários em cada linha da tela)?

Experimente este:

```
10 PRINT "RENATO";
20 GOTO 10
```

Qual foi o resultado?

Note que este programa tem duas coisas diferentes do outro:

1. existe um espaço em branco entre o fim do nome (letra O) e a última aspa.
2. existe um ; no final da linha 10.

Para explicar o que aconteceu é fácil:

colocando um ; no final da linha 10, a próxima vez que o computador repeti-la, o nome aparecerá unido ao anterior.

Lembra-se de que o ponto e vírgula serve para separar as partes de uma instrução PRINT, que queremos que apareçam unidas?

Aqui também funciona deste jeito.

Se eu não puser um ; no final da linha PRINT, a próxima vez que o computador obedecer outro PRINT, ele passará a escrever na linha de baixo, e assim por diante.

Coloquei o espaço em branco depois do nome RENATO para não aparecer assim na tela:

RENATORENATORENATORENATO-
RENATORENATO etc.

Experimente colocar o seu nome ou então com uma frase assim:

10 PRINT "MARIO E O MAIORAL";

ou então:

10 PRINT "JOÃOZINHO QUER NAMORAR MARIA";

Agora você vai ver que o computador TK-85 também pode ser usado para desenhar uma porção de coisas na tela, através da programação.

Essa maneira de desenhar gráficos especiais é chamada de MODO GRÁFICO.

E ele funciona de uma maneira muito simples: é também com a instrução PRINT.

Se você olhar bem o teclado, verá que na parte de cima de algumas teclas existem uns desenhos engraçados.

Eles são chamados de CARACTERES ou BLOCOS GRÁFICOS.

Nós temos no teclado os seguintes blocos:

EXERCÍCIOS

1. Faça um programa para colocar na tela as palavras TIC e TAC, uma em cada linha, até encher toda a tela, assim:

Resposta:

2. Modifique o programa para escrever toda a tela, mas com as palavras TIC e TAC unidas.

Resposta:

3. O que faz este programa?

```
110 PRINT TAB 16; "I"
125 GOTO 110
```

Resposta:

4. O que há de errado com este programa?

```
40 PRINT "PROGRAMA BASIC";
55 GOTO 45
```

Que mensagem de erro vai aparecer, se você colocar esse programa no computador e executá-lo?

5. O que faz este programa?

```
100 GOTO 100
```

Como devo fazer para interromper o programa, neste caso?

Se você prestar bem atenção, poderá ver que eles correspondem aos pedacinhos de um quadrado dividido em quatros partes, combinados de diversas maneiras:

Alguns dos blocos gráficos também têm partes cinza (pontinhos bem pequenos e próximos).

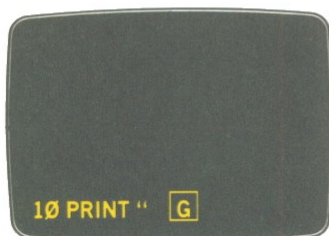
Com estes blocos gráficos podemos montar uma série de desenhos interessantes:

Veja como é fácil!

Para colocar dentro de uma instrução PRINT os blocos gráficos, eles devem estar dentro de aspas. Coloque a seguinte linha de programa (pressione NEW antes):

Para poder digitar os blocos gráficos, devemos mudar o cursor L (letras) para G (gráficos).

Isto é feito pressionando-se simultaneamente as teclas SHIFT e 9. Veja que está escrita a palavra GRAPHICS em cima do rótulo da tecla 9. Isso quer dizer GRÁFICOS em português, certo?

Note que, quando você fizer isso, o cursor muda para G:

Agora tudo o que você digitar pelo teclado aparecerá de forma diferente.

Pressionando AO MESMO TEMPO a tecla SHIFT e uma tecla contendo um bloco gráfico, ele aparecerá na tela.

Experimente digitar SHIFT e a tecla W:

Depois SHIFT 6, SHIFT W e SHIFT 6 de novo:

Veja que está se formando uma figura, porque os blocos gráficos ficam grudados (o que não acontece com as letras).

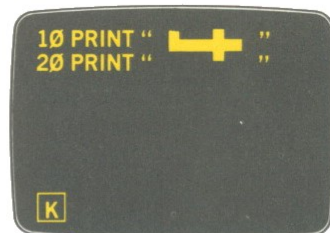
Para colocar as aspas no final, temos que sair do MODO GRÁFICO.

Apertamos então as teclas SHIFT e 9 novamente:

Veja que o cursor mudou de novo para L.

Agora sim pressione aspas e depois NEW LINE.

Pressione RUN para ver o que acontece:

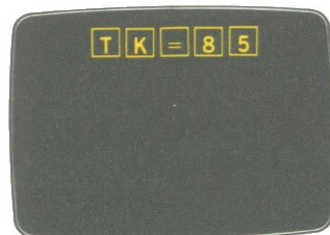
Colocando várias linhas de programa alinhadas, os blocos gráficos das linhas de cima se grudam com os de baixo:

Veja que agora conseguimos fazer um aviãozinho.

Tente fazer, como exercício, as outras figuras mostradas logo no começo desta lição!

O modo gráfico também pode ser usado para escrever as letras e outros caracteres em cor preta sobre o fundo branco, ao invés de como aparecem normalmente na tela.

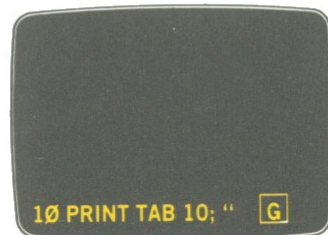
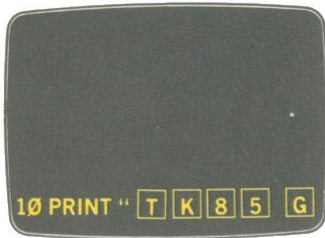
Esta maneira de escrever se denomina MODO INVERSO, ou caracteres em campo invertido.

Vamos dar um exemplo:

Suponha que eu quero escrever o nome deste computador bem no meio do topo da tela, em caracteres inversos. Assim:

Para conseguir isso, você deve usar o MODO GRÁFICO para escrever o nome. Só que não deve pressionar a tecla SHIFT antes, como no caso dos blocos gráficos. Pressione as teclas das letras que você quiser colocar normalmente:

Comece digitando:

10 PRINT TAB 10;"

Depois pressione SHIFT junto com a tecla 9 para mudar o cursor para G (GRAPHICS):

Agora pode digitar as letras TK 85:

Viu como saiu direitinho?

Agora pressione SHIFT e 9 juntos novamente, para mudar o cursor para L, e digite o sinal de aspas do final e pressione NEW LINE:

Pressione RUN e veja o que acontece!

EXERCÍCIOS

1. Faça de novo o exercício de escrever TIC e TAC na tela, com um programa (lembra-se? TIC em uma linha e TAC em outra). Só que desta vez TIC deve aparecer em caracteres normais (letras brancas em fundo preto) e TAC em caracteres inversos (letras pretas sobre fundo branco).
Resposta:

2. Faça um programa para colocar no topo da tela uma moldura assim, com seu nome no meio dela. Experimente com diversos caracteres gráficos até achar uma moldura bem bonita:
Resposta:

3. Vamos fazer um programa para desenhar uma tapeçaria na tela. É muito fácil:
Coloque em primeiro lugar uma instrução PRINT contendo 7 ou 9 blocos gráficos diversos, um grudado no outro (entre aspas). Depois coloque uma instrução GOTO para repetir este PRINT.
Pressione RUN e aprecie o resultado!

Coloque diferentes combinações de blocos gráficos para testar qual vai ser a tapeçaria mais bonita. Escreva o programa que produziu a mais bonita aqui:

TESTE

7.1. O que significa para o computador quando ele está mostrando na tela os seguintes cursores:

K

L

F

G

7.2. Defina o que quer dizer:

MODO GRÁFICO

MODO INVERSO

7.3. Que teclas devemos apertar para conseguir digitar os blocos gráficos (cursor G)?

7.4. Faça um programa para desenhar na tela a figura de uma cabeça de palhaço dando risada.

7.5. Modifique o programa anterior para colocar a cabeça do palhaço no lado direito da tela (dica: use a função TAB em todas as linhas PRINT).

7.6. Por que eu devo fazer o cursor voltar a ser L antes de digitar as aspas no final de uma instrução PRINT com blocos gráficos?

7.7. Vamos rememorar. Para que servem as teclas:

NEW LINE

RUBOUT

RUN

NEW

⇐

⇒

GRAPHICS

FUNCTION

SHIFT

BREAK

SPACE

CONT

E para que servem as instruções e funções (dê um exemplo de cada).

PRINT

TAB

SQR

sando Variáveis dentro de um Programa

Agora que você percebeu a utilidade das variáveis para armazenar dados e resultados intermediários, vamos ver alguns exemplos com programas. Digite no computador cada um dos programas abaixo, e veja o resultado. Procure substituir alguns valores para obter diferentes resultados.

1. Dados dois números, A e B, faça um programa para obter o resultado da soma, subtração, multiplicação e divisão entre os dois.

SOLUÇÃO

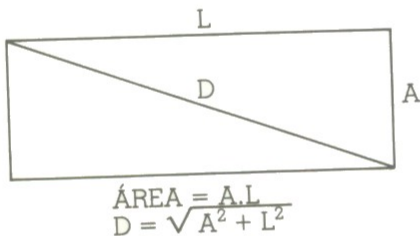
Note que das linhas 30 a 60 existe uma diferença entre o que está entre aspas (é uma mensagem) e o que está fora das aspas (é uma expressão). O que é mensagem é mostrado na tela exatamente como está. O que é expressão é usado para calcular.

```
10 LET A=25
20 LET B=12
30 PRINT"A + B =";A+B
40 PRINT"A – B =";A–B
50 PRINT"A × B =";A*B
60 PRINT"A / B =";A/B
```

Pergunta: Qual é a vantagem de se colocar primeiro os dados (no caso, 25 e 12) dentro de variáveis?

2. Dados o lado (L) e a altura (A) de um retângulo, faça um programa para calcular a área e o comprimento da diagonal.

ÁREA = A.L

$D = \sqrt{A^2 + L^2}$

SOLUÇÃO

Temos que usar as fórmulas correspondentes ao cálculo da área e da diagonal de um retângulo (veja a figura).

Em BASIC, a exponenciação é obtida com o operador ** (ele tem uma tecla especial: não se pode apertar a tecla * duas vezes, pois haverá erro no resultado. A tecla ** fica acima da letra H). A raiz quadrada é obtida com a função SQR: para digitá-la, pressione simultaneamente SHIFT e NEW LINE, depois pressione a tecla da letra H.

```
10 LET L=10
20 LET A=5
30 PRINT"PROBLEMA DO RETÂN-
GULO"
40 PRINT"= = = = = = = = = = ="
50 PRINT
60 PRINT"O COMPRIMENTO DO LA-
DO =";L
70 PRINT"A ALTURA =";A
80 PRINT
90 LET ÁREA = L*A
100 LET DIAG = SQR L**2 + A**2
110 PRINT"ÁREA =";ÁREA
120 PRINT"DIAGONAL =";DIAG
```

Pergunta: As linhas 90 e 100 são necessárias? Por quê?

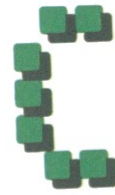
EXERCÍCIOS

1. Refaça os seguintes exercícios do capítulo 3, usando variáveis:

1.1. A altura do King Kong é igual a 37 pés e 8 polegadas. Quantas polegadas e quantos metros ele tinha de altura?
Resposta:

1.2. O jogador Zico tinha 300 000 dólares no banco. Depois recebeu como luvas metade de 456 784 dólares, menos o imposto de 40 000 dólares. Quantos cruzeiros ele terá no total (faça a conta como se o dólar valesse 2 800 cruzeiros. Repita o cálculo como se o dólar valesse 6 000 cruzeiros).
Resposta:

1.3. Um carro tinha 25.6 litros de combustível no tanque. Depois de uma viagem ficou apenas com 17.3. Quantos litros foram gastos? Quanto custou a viagem, se o litro de gasolina custa 1.000 cruzeiros? Quanto ele economizaria em litros e cruzeiros se o carro gastasse 20% menos?
Resposta:

1.4. A idade de uma pessoa é igual a A anos, M meses e D dias. Calcule quantos meses, dias e horas esta pessoa já viveu, aproximadamente.
Resposta:

aixinhas para Nomes

Até agora vimos que a memória do computador pode ser usada para armazenar números. Esses números podem ser o resultado de uma expressão matemática:

```
LET C = 23 + (25.7-14.2)/13
```

Neste exemplo, C é o nome da "caixinha" ou memória que receberá o resultado da conta, à direita do sinal de igualdade.

Outra maneira de colocar um valor numérico em uma variável é por meio de uma constante:

```
LET PI = 3.1415
```

Neste exemplo, PI é o nome da variável. O número à direita do sinal de igualdade não contém nenhuma operação matemática ou nome de variável, por isso é chamado de constante.

Mas não são só números que o computador consegue armazenar.

PODEMOS GUARDAR LETRAS E OUTROS CARACTERES NA MEMÓRIA DO COMPUTADOR

Para isto, o BASIC do seu computador nos permite usar "caixinhas" especiais, que servem para guardar na memória conjuntos de letras e de outros caracteres.

UM CONJUNTO DE CARACTERES É CHAMADO DE CADEIA

Outro nome muito usado para descrever um conjunto de caracteres é "string", uma palavra inglesa que significa "cordão".

Você já viu antes cadeias de caracteres, lembra-se?

Por exemplo, no comando:

```
PRINT "RENATO"
```

Neste caso, a cadeia é uma palavra formada de 6 caracteres:

R, E, N, A, T e O.

Mas uma cadeia pode incluir outros caracteres. Pode ser uma frase:

```
PRINT "O BRASIL TEM 120 MI-
LHÕES DE HABITANTES"
```

ou, então, pode ser uma mistura de caracteres:

```
PRINT "+% *** =8"
```

ou caracteres gráficos:

```
PRINT "▙"
```

Como podemos dizer ao computador que queremos armazenar uma cadeia de caracteres em uma variável da memória?

Preste atenção:

LET N$ = "RENATO"

Este comando LET está dizendo para o computador:

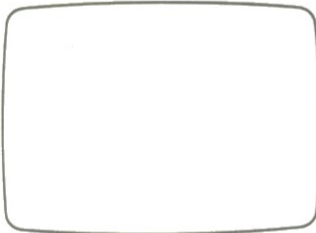
GUARDE A CADEIA DE CARACTERES (ENTRE ASPAS) NA VARIÁVEL CHAMADA N$

O sinal de cruzeiro ($), também chamado de cifrão, serve para indicar ao computador que a variável é do tipo alfabético (ou seja, deverá armazenar letras e outros caracteres).

Note que, se você esquecer de pôr o sinal $ depois do nome da variável, o computador vai "estrilar"!

Experimente digitar no computador:

LET N = "RENATO"

Veja o que aparece na tela:

Experimente fazer o mesmo, na forma de um programa, assim:

10 LET N = "RENATO"

e depois digite RUN. O que acontece?

A mensagem de erro está indicando que os tipos não combinam, ou seja:

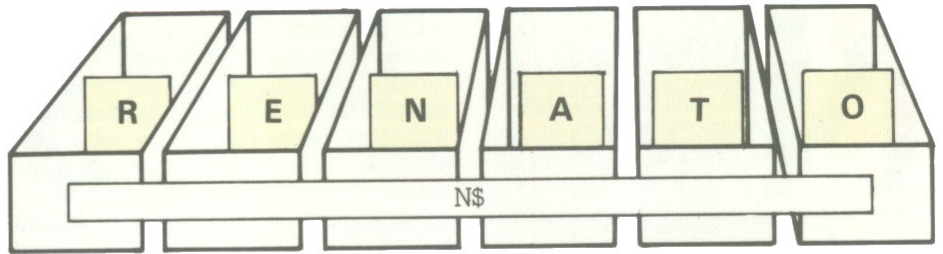
- a variável à esquerda de = é do tipo numérico
- a constante à direita de = é do tipo alfabético

O mesmo erro ocorreria se você tentasse o contrário:

10 LET N$ = 123

O que está errado nesta linha?

Como o Computador Armazena um Nome?

No começo deste capítulo, vimos que o computador usa unidades de

memória (as caixinhas mostradas nos desenhos) para armazenar números.

Para armazenar nomes, a coisa é um pouco mais complicada, mas também é fácil de ser entendida.

Vejamos: a memória do microcomputador é feita de tal forma que comporta apenas um caractere (uma letra, por exemplo) em cada caixinha. Assim, o nome RENATO seria guardado em 6 caixinhas agrupadas que receberiam a "etiqueta" (nome da variável) N$.

O computador permite que tenhamos até 255 caixinhas agrupadas para guardar uma cadeia de caracteres, todas recebendo um único nome de variável.

Assim, podemos armazenar cadeias de caracteres de qualquer comprimento (até um máximo de 255), sem nos preocupar com quantas caixinhas o computador vai usar. Isto ele faz automaticamente.

LET A$ ="NABUCODONOSOR ERA UM REI ASSIRIO CRUDELISSIMO"

LET B$ ="OLA!"

LET C$ ="   "

(O que diz este último comando?)

O IMPORTANTE É COLOCARMOS OS CARACTERES ENTRE ASPAS

Para descobrir que a cadeia de caracteres está armazenada em uma variável alfabética, podemos usar o comando PRINT. Por exemplo, digite o seguinte programa no computador:

```
10 LET NOME$ ="MICKEY MOUSE"
20 PRINT NOMES$;"ERA UM RATO!"
```

Veja o que vai acontecer, se você der o comando RUN:

Vamos entender juntos este programa, linha a linha:

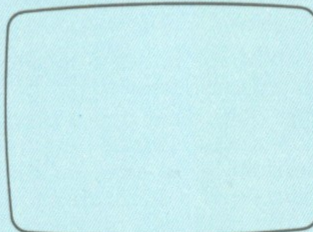
Linha 10 ⇒ armazena a constante "MICKEY MOUSE" na variável NOME$

Linha 20 ⇒ mostra na tela o conteúdo de NOME$, seguido da constante alfabética "ERA UM RATO!", ambos na mesma linha.

EXERCÍCIOS

1. Escreva abaixo o comando que você daria ao computador, para armazenar o nome do seu time de futebol na variável chamada TIME$.

2. O que o computador mostrará na tela com este programa?

```
10 LET FLOR$ ="ROSA"
20 PRINT"UMA"; FLOR$;"É UMA";
FLOR$
```

E o que vai acontecer, se eu trocar a linha 10 por:

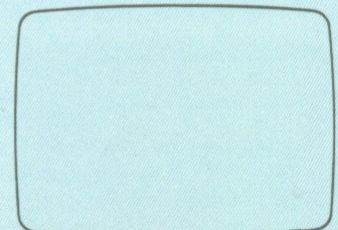
```
10 LET FLOR$ ="MARGARIDA"
```

3. E este programa, o que vai mostrar na tela?

```
10 LET PLANETA$ ="MARTE"
20 LET PLANETA$ ="JÚPITER"
30 PRINT PLANETA$
```

Explique por quê:

4. Agora diga o que aparecerá na tela com este programa:

```
10 LET COMIDA$ ="ARROZ"
20 LET ALMOÇO$ ="COMIDA$
30 PRINT"NO ALMOÇO VAI TER";
ALMOÇO$
```

NÚMEROS E NOMES PELO TECLADO

Até agora, aprendemos a armazenar números (e nomes), dentro de um programa, usando o comando LET.

Vamos fazer, por exemplo, um programa que armazene a idade e o nome de uma pessoa, em duas variáveis, e depois calcule quantos dias ela viveu, aproximadamente.

Este programa poderia ser assim:

```
10 LET ANOS = 14
20 LET NOME$ ="ROBERTO"
30 PRINT NOME$;"VIVEU MAIS
DE"; ANOS * 365;"DIAS"
```

Depois de digitar este programa no computador:

Damos o comando RUN, para ver o resultado:

O programa é muito interessante, mas há um problema.

O que devemos fazer se quisermos colocar a idade e o nome de uma outra pessoa para fazer o mesmo cálculo?

Você já sabe a resposta:

SEM ALTERAR A LINHA 30, DIGITAMOS DE NOVO AS LINHAS 10 E 20

Assim, por exemplo:

```
10 LET ANOS = 37
20 LET NOME$ ="O PROFESSOR
RENATO"
```

O programa agora vai mostrar na tela:

O problema é pequeno, é verdade, mas, se a classe toda e todos os amigos do seu irmãozinho quiserem fazer a mesma coisa, vai dar muito trabalho. Deve haver um jeito mais fácil do que ficar sempre mexendo no programa!

Felizmente, este jeito existe: é o comando INPUT.

Este comando substitui o LET, permitindo que um valor (número ou nome) seja fornecido ao programa pelo teclado, sem haver necessidade de modificar o programa.

Preste bem atenção como o programa acima seria modificado:

```
10 INPUT ANOS
20 INPUT NOME$
```

A linha 30 ficaria igual:

O comando INPUT na linha 10 significa a seguinte ordem para o computador:

ESPERE ALGUÉM ESCREVER UM NÚMERO NO TECLADO, E GUARDE-O NA VARIÁVEL ANOS

O comando INPUT na linha 20 significa:

ESPERE ALGUÉM ESCREVER UM NOME NO TECLADO, E GUARDE-O NA VARIÁVEL NOME$

Vamos ver agora qual o resultado deste programa. Digite RUN:

O computador parou e mostrou na linha de baixo o cursor [L]

Isto não quer dizer que o programa não funcionou (senão apareceria uma mensagem de erro, ou o cursor K), significa que o computador encontrou a linha 10, e está obedecendo ao comando INPUT, ou seja, parou para que você digite um número.

Tudo o que você digitar vai aparecer na linha onde está o cursor [L]

Depois de ter digitado o número que deseja, pressione NEW LINE ou ENTER.

Note que aparece de novo o cursor L na linha de baixo, agora entre aspas.

Isto significa que o computador deve estar esperando a entrada de uma cadeia de caracteres, na linha 20 do programa. Digite um nome:

e pressione a tecla NEW LINE. Agora o programa guardou o nome na variável correspondente, e vai para a linha 30, fazendo o cálculo e mostrando o resultado:

Entendeu?

Para repetir o programa, basta digitar de novo a tecla RUN, e tudo se repete. Tente colocar várias idades e nomes para ver o que acontece.

Melhor ainda, adicione ao programa acima a linha:

```
40 GOTO 10
```

Desta forma, o programa ficará se repetindo indefinidamente. Depois de entrar a idade e o nome, ele vai mostrar o resultado na tela, e pedir nova idade, e assim por diante:

ARNALDO VIVEU MAIS DE 4380 DIAS

JOAQUIM VIVEU MAIS DE 5110 DIAS

FERNANDA VIVEU MAIS DE 6570 DIAS

O programa só vai parar quando a tela ficar cheia. Neste caso, aparece a seguinte mensagem, na última linha da tela:

5/30

que quer dizer: PAREI COM A TELA CHEIA NA LINHA 30

Para continuar o programa, basta pressionar a tecla marcada com o comando CONT (continue), que limpa a tela e reinicia tudo de novo na primeira linha.

Como parar o programa antes que a tela fique cheia?

Experimente pressionar a tecla BREAK.

Não funciona quando o cursor [L] está aparecendo, certo?

O único jeito é pressionar a tecla marcada STOP, e, em seguida, a tecla NEW LINE.

Se o cursor [L] estiver entre aspas, só pressionar STOP não resolverá.

É necessário antes apagar as aspas da esquerda, para que o computador não pense que o STOP é apenas uma cadeia de caracteres.

Dialogando com o Computador

Os comandos INPUT e PRINT permitem escrever programas de CONVERSAÇÃO, ou seja, uma espécie de "diálogo" entre o computador e o usuário.

Veja, por exemplo, este simpático programa:

```
10 PRINT"OLA, EU SOU O MICRO-
COMPUTADOR DE QUE MEU DONO
TANTO FALA"
20 PRINT
30 PRINT"QUAL E O SEU NOME?"
40 INPUT N$
50 CLS
60 PRINT"MUITO PRAZER,"; N$
70 PRINT"TENHO CERTEZA DE QUE
VOCÊ VAI GOSTAR DE MIM"
```

Na tela o diálogo apareceria assim:

Você pode imaginar muitos "diálogos" como este.

Experimente!

Estruturando o Programa

Sabendo os comandos INPUT, LET e PRINT, você já consegue fazer muitas coisas com o computador. Todos os problemas que têm a seguinte forma:

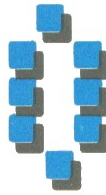
— fornecer dados conhecidos para um problema

— procurar o resultado a partir destes dados, usando uma ou mais fórmulas

— mostrar o resultado

podem ser programados facilmente em BASIC, usando o computador.

Nesta seção, vamos aprender como estruturar um programa deste tipo, e vamos programar vários exemplos e exercícios, para fixar bem a maneira de fazê-lo.

Você poderá aplicar a mesma receita para programas que você mesmo vai bolar depois.

O comando REM

Uma coisa muito importante para quem deseja aprender a fazer bons programas é a documentação. O que é isso?

À medida que os programas que você faz vão ficando cada vez mais compridos, começa a ficar difícil entendê-los. Mesmo que tenha sido você que os escreveu, depois de alguns meses pode acontecer que não se lembre mais para que servem algumas partes do programa.

A linguagem BASIC tem um comando cuja única função é ajudar o programador a colocar explicações em linguagem corrente dentro de um programa. Essas explicações são chamadas de DOCUMENTAÇÃO INTERNA DE UM PROGRAMA.

Esse comando se chama REM.

REM significa REMARK, em inglês, que quer dizer COMENTÁRIO.

O comando REM não tem nenhuma outra serventia além de colocar linhas escritas, com alguma explicação ou comentário, dentro de um programa. Portanto, ele não é um comando EXECUTÁVEL, ou seja, não é executado pelo computador, como os comandos LET e PRINT, por exemplo.

O comando REM é escrito da seguinte maneira:

23 REM COMENTÁRIO

Depois do comando REM, na mesma linha, podemos colocar o que quisermos (inclusive caracteres gráficos).

O comando REM é útil para muita coisa:

— identificar o nome do programa, quem o escreveu, em que data etc. Esta parte do programa geralmente é colocada no começo, e é chamada de CABEÇALHO DO PROGRAMA. Quando você tiver muitos programas gravados em fita, você saberá como esse cabeçalho é necessário para evitar confusões...

— verificar para que servem as variáveis dentro do programa

— identificar as diferentes seções de um programa

— colocar linhas de separação entre as partes de um programa

Todos esses recursos são úteis para facilitar a leitura e o entendimento de um programa, tanto para você como para os outros. Você entenderá melhor a utilização do REM no programa abaixo:

PROBLEMA

Fazer um programa para converter dólares em cruzeiros, à taxa do dia:

SOLUÇÃO

A maneira mais fácil e direta de fazer este programa seria:

```
20 INPUT TAXA
40 INPUT VALOR
60 LET CRUZ = TAXA * VALOR
80 PRINT CRUZ
90 GOTO 40
```

Como você percebe, o programa solicita apenas uma vez, na linha 20, a taxa atual do dólar (e a armazena na variável TAXA). Depois solicita a entrada do valor em cruzeiros (e a armazena na variável VALOR, na linha 40). Finalmente, calcula, na linha 60, e mostra o resultado, em cruzeiros, na linha 80. A linha 90 faz o programa perguntar o novo valor em cruzeiros, para ser convertido, e assim por diante, até pressionarmos a tecla STOP.

Este programa é bem simples, mas tem as três partes fundamentais de qualquer programa de resolução de problemas:

— ENTRADA (linhas 20 e 40)

— CÁLCULO (linha 60)

— SAÍDA (linha 80)

Note que a pessoa que for usá-lo precisa saber para que ele serve, pois o programa não dá nenhuma "dica" sobre isto, sobre que valores devem ter entrada, e quando, etc.

Por isso, um programa realmente útil tem comandos com mensagens de instruções para o usuário, e verificação dos resultados. Veja como fica muito melhor assim:

```
10 PRINT"TAXA ATUAL DO DOLAR?"
20 INPUT TAXA
30 PRINT"VALOR EM DOLARES?"
40 INPUT VALOR
60 LET CRUZ = TAXA * VALOR
80 PRINT"VALOR EM CRUZEIROS =";
CRUZ
90 GOTO 30
```

Veja que o programa melhora bastante, pois informa ao usuário que valores ele tem que digitar, e depois exprime o resultado com uma mensagem de esclarecimento. Na tela fica assim:
para completar o nosso programa, vamos colocar todos os comentários com os comandos REM, documentando-o adequadamente:

```
1 REM *********************
2 REM * PROGRAMA CONVERTE *
3 REM * R.SABBATINI *
4 REM * 12.SET.84 *
5 REM *********************
6 REM
7 REM *** ENTRADA DE DADOS ***
8 REM
10 PRINT"TAXA ATUAL DO DÓLAR?"
20 INPUT TAXA
25 PRINT TAXA
30 PRINT"VALOR EM DÓLARES?"
40 INPUT VALOR
45 PRINT VALOR
54 REM
50 REM *** CÁLCULO ***
55 REM
60 LET CRUZ = TAXA * VALOR
64 REM
65 REM *** RESULTADO ***
66 REM
70 PRINT
80 PRINT"VALOR EM CRUZEIROS =";
CRUZ
90 GOTO 30
```

Note que foram adicionadas as seguintes linhas:
1 a 5 ◊ cabeçalho
6 a 8, 54 a 55 e 64 a 66 ◊ identificação das seções
25 e 45 ◊ mostram na tela os valores de entrada, para melhor referência

O programa acima é bastante legível e de fácil modificação, mesmo por quem não o desenvolveu. As seções de entrada, cálculo e saída são facilmente identificáveis.

Conclusões sobre o REM

O comando REM é muito útil, como você viu.

Entretanto existem algumas coisas de que precisamos lembrar, quando formos usá-lo:

1. O comando REM não é uma ordem para o computador. Tudo o que estiver escrito nele (inclusive comandos válidos em BASIC) fica como que invisível para o computador e visível só para você.

2. Os comandos REM ocupam espaço de memória no computador. Se o programa for muito longo e não couber na memória, é necessário limitar o número de REMs ou tirá-los.

3. Se você vai desenvolver um programa muito longo, tome como hábito ir escrevendo os comentários REM em seu interior à medida que vai digitando as linhas de comandos.

4. Os comentários REM não são a única forma de documentação. Depois veremos outras.

TESTE

8.1. Defina:
VARIÁVEL ....
CONSTANTE ....
VARIÁVEL NUMÉRICA ....
VARIÁVEL ALFABÉTICA ....
CADEIA DE CARACTERES ....

8.2. Qual é a vantagem principal de se utilizar variáveis em um programa BASIC?
....
....
....

8.3. Para que servem os comandos:
LET ....
INPUT ....
REM ....

8.4. Marque com V se a afirmativa for verdadeira, e com F, se for falsa:
( ) o comando REM é interpretado e executado pelo computador
( ) os comandos REM de um programa não ocupam a memória
( ) o comando INPUT pode ser usado sem uma variável
( ) o comando INPUT pode ser usado para entrada de números e nomes
( ) quando o computador encontra um comando INPUT, interrompe o processamento de um programa e fica esperando uma entrada
( ) o comando INPUT solicita dados pelo teclado
( ) o comando LET serve apenas para fazer cálculos

8.5. Assinale quais os nomes de variáveis que são corretos (C) e errados (E) e explique por quê:
TK 85
1983
ABCD123
COISA$
COI$SA
XZ/3
4DADOS

8.6. Que resultado vai mostrar o computador no programa abaixo? (Explique por que, para cada linha PRINT):

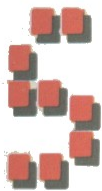
```
10 LET A=125
20 LET B=25
30 LET C=A
40 PRINT A,B,C
50 LET C=A+B
60 LET D=A
60 LET D=D+1
70 PRINT A,C,D
```

8.7. Escreva um programa que pergunte o seu nome e depois encha a tela com ele.

8.8. Faça um programa para entrada de três notas pelo teclado, e depois calcule a soma e a média simples das notas. Coloque também o nome do aluno.

8.9. Qual é a vantagem principal de se usar o comando INPUT em um programa em BASIC?

8.10. Cite três utilidades do comando REM em um programa:
1.
2.
3.

8.11. Quais são as três partes principais de um programa para resolver um problema de cálculo?
1.
2.
3.

SOLUÇÃO DOS PROGRAMAS PROPOSTOS

Abaixo, a solução dos programas propostos no capítulo 5 deste livro.

TESTE

PÁGINA 21

6.12. Faça um programa para colocar na tela uma lista numerada de três times de futebol. Por exemplo:

1 PALMEIRAS
2 BOTAFOGO
3 FLAMENGO

Resposta:

```
10 PRINT "1", "PALMEIRAS"
20 PRINT "2", "BOTAFOGO"
30 PRINT "3", "FLAMENGO"
40 STOP
```

6.13. Faça um programa para colocar um sinal + em cada canto da tela (dica: use o comando PRINT AT).

Resposta:

```
10 PRINT AT 1,1, "+"
20 PRINT AT 1,31, "+"
30 PRINT AT 20,1, "+"
40 PRINT AT 20,31, "+"
50 END
```

EXERCÍCIOS

PÁGINA 23

1. Faça um programa para colocar na tela as palavras TIC e TAC, uma em cada linha, até encher toda a tela.

Resposta:

```
10 PRINT "TIC"
20 PRINT "TAC"
30 GOTO 10
```

2. Modifique o programa acima para escrever em toda a tela, mas com as palavras TIC e TAC unidas.

Resposta:

```
10 PRINT "TIC"
20 PRINT "TAC"
30 GOTO 10
```

TESTE

PÁGINA 30

8.7. Escreva um programa que pergunte o seu nome, e depois encha a tela com ele.

Resposta:

```
10 PRINT "QUAL É O SEU NOME?"
20 INPUT NOME $
30 CLS
40 PRINT NOME $; " ";
50 GOTO 40
```

8.8. Faça um programa para entrada de três notas pelo teclado, e depois calcule a soma e a média simples das notas. Coloque também o nome do aluno.

Resposta:

```
10 REM *********************
20 REM * MEDIA DE 3 NOTAS *
30 REM *********************
40 REM
50 REM *** ENTRADA DAS NOTAS ***
60 REM
65 PRINT "CALCULO DE MEDIA"
68 PRINT
70 PRINT "ENTRE 3 NOTAS:"
80 INPUT N1
90 PRINT "NOTA 1 = "; N1
100 INPUT N2
110 PRINT "NOTA 2 = "; N2
115 INPUT N3
120 PRINT "NOTA 3" = ; N3
130 REM
140 REM *** CALCULA SOMA E
       MEDIA ***
150 REM
160 LET SOMA = N1 + N2 + N3
170 LET MEDIA = SOMA/3
180 REM
190 REM ***MOSTRA RESULTADO***
200 REM
210 PRINT
220 PRINT "SOMA = "; SOMA
230 PRINT "MEDIA = "; MEDIA
240 STOP
```

Exemplo de execução:

1

2

CALCULO DA MEDIA ENTRE
TRES NOTAS:
8 L

digite a 1.ª nota

3

CALCULO DA MEDIA ENTRE
TRES NOTAS:
NOTA 1 = 8
L

4

CALCULO DA MEDIA ENTRE
TRES NOTAS:
NOTA 1 = 8
9 L

digite a 2.ª nota

5

CALCULO DA MEDIA ENTRE
TRES NOTAS:
NOTA 1 = 8
NOTA 2 = 9
L

6

digite a 3.ª nota

7

CALCULO DA MEDIA ENTRE
TRES NOTAS:
NOTA 1 = 8
NOTA 2 = 9
NOTA 3 = 4

CALCULO DA MEDIA ENTRE
TRES NOTAS
NOTA 1 = 8
NOTA 2 = 9
NOTA 3 = 4
SOMA = 21
MEDIA = 7
9/24Ø

resultados

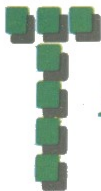



TABELA DE CÓDIGOS INDICADORES DE ERROS

Apresentamos abaixo uma tabela completa com todos os códigos indicadores de erros que podem aparecer durante a digitação ou a execução de um programa. Lembre-se que este código sempre aparece na linha de baixo da tela, mostrando também o número da linha em que ele ocorreu. Exemplo:

5/15

Significa: ocorreu o erro número 5, na linha 15.
O erro número 5 significa: não há mais espaço na tela para continuar a listagem. Na coluna SITUAÇÕES você poderá ver com quais comandos pode ocorrer este erro. No nosso exemplo, pode ocorrer com PRINT, LIST etc.
Algumas vezes, a tabela também indica o que fazer para sair do erro. No caso do erro número 5, você deverá digitar o comando CONT para limpar a tela.

CÓDIGO	SIGNIFICADO	SITUAÇÕES
0	Não houve erro. Execução bem sucedida ou salto para uma linha de maior número que qualquer outra existente.	Qualquer
1	A variável de controle não existe (não foi estabelecida por uma instrução FOR), mas existe uma outra variável com o mesmo nome.	NEXT
2	A linha indicada usa uma variável não definida previamente por uma instrução LET ou INPUT. Para matrizes isso acontecerá se estas forem usadas antes de ser dimensionadas pela instrução DIM. E, para uma variável de controle, isso ocorrerá caso ela for usada antes de ter sido definida como variável de controle por uma instrução FOR, e se não houver nenhuma variável simples com o mesmo nome.	Qualquer
3	A matriz não contém o elemento especificado. Caso o índice esteja fora de faixa (isto é, seja negativo ou acima de 65535), ocorrerá o erro B.	Matrizes
4	Espaço insuficiente na memória. Observe que o número de linha na indicação (após o /) poderá estar incompleto na tela, devido à falta de memória; assim, por exemplo, 4/20 poderá surgir como 4/2.	LET, INPUT, DIM, PRINT, LIST, PLOT, UNPLOT, FOR, GOSUB. Às vezes, durante a avaliação de funções.

CÓDIGO	SIGNIFICADO	SITUAÇÕES
5	Não há mais espaço na tela. CONT criará espaço, limpando a tela.	PRINT, LIST, PLOT, UNPLOT.
6	Resultado fora dos limites do computador. Os cálculos produziram um número superior a 10^{38}.	Qualquer cálculo aritmético.
7	RETURN sem um GOSUB correspondente.	RETURN
8	Você tentou um comando INPUT não permitido.	INPUT
9	Instrução STOP executada. CONT fará a execução começa da próxima linha.	STOP
A	Argumento inválido para certas funções.	SQR, LN, ASN, ACS
B	Número inteiro fora da faixa (—655 34 a 65535).	RUN,RAND, POKE, DIM, GOTO, GOSUB, LIST, PAUSE, PLOT, UNPLOT, CHR$, PEEK, USR. Quando de um acesso a uma matriz
C	O texto do argumento (de string) de VAL não forma uma expressão numérica válida.	VAL
D	Programa interrompido por: 1. Tecla BREAK pressionada ou 2. Um STOP foi digitado no teclado.	Ao fim de qualquer instrução ou em LOAD, SAVE, PRINT, LIST ou COPY.
E	Não utilizado	
F	O nome fornecido para o programa está em branco.	

RESUMO

Vamos rever agora todos os comandos e instruções que você aprendeu até aqui, em ordem alfabética, para facilitar o seu trabalho posterior.

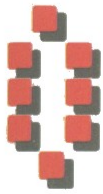
Na relação abaixo,
V representa uma variável;
n representa expressão numérica que é arrendondada para o número inteiro mais próximo;
e representa uma expressão.

Note que expressões arbitrárias são permitidas em qualquer lugar, exceto para número de linha no início da instrução.
Todas as instruções, exceto INPUT, podem ser usadas tanto em linhas imediatas como em linhas de comando.

CONT	Continua a execução após um BREAK ou STOP. Normalmente é digitado pelo teclado (não é colocado dentro de um programa).
CLS	Limpa a tela.
GOTO *n*	Salta para linha *n* (ou, se não existe, para a que vem depois dela). Exemplo: GOTO 2Ø
INPUT *V*	Pára e espera que seja feita uma entrada de dados. INPUT não pode ser usado diretamente pelo teclado, pois ocorrerá erro 8. Se o primeiro caractere em uma linha de INPUT é STOP, o programa pára com denotação D. Exemplo: INPUT A$.
LET *V*=*e*	Atribui o valor *e* à variável *v*. LET não pode ser omitido. Uma variável simples é indefinida, até figurar em uma instrução LET ou INPUT. Exemplo: LET X = A + 25/3
LIST	*Lista o programa na tela, começando na 1ª linha.*
LIST *n*	Lista o programa na tela, começando na linha *n*. Ocorrem erros 4 ou 5, se a listagem for muito longa para a tela. Exemplo: LIST 2Ø
NEW	Apaga o programa e todas as variáveis, para poder colocar outro na memória.
PRINT	Mostra algo na tela (o que estiver indicado depois da palavra PRINT), e que é chamado *item de impressão*. O item pode ser: (1) Vazio, isto é, nada aparece depois. Neste caso é mostrada na tela uma linha em branco. Exemplo: PRINT (2) Uma expressão numérica. É mostrado o resultado da expressão. Exemplo: PRINT 25 + 14 O resultado é impresso em notação decimal ordinária de até 8 dígitos significativos e sem zeros após o ponto decimal. Um ponto decimal no início é sempre seguido de um zero. (3) Uma expressão entre aspas. É mostrado exatamente o que aparece. Exemplo: PRINT "BOM DIA". (4) AT m,n A posição de impressão é mudada para linha m (contando do topo), coluna n (contando da esquerda). Se |m| = 22 ou 23, resulta erro 5 (5) TAB n

A posição de impressão é movida para coluna n, permanecendo na mesma linha, a menos que isso envolva retorno na mesma linha; neste caso, move-se para a próxima linha. Exemplo: PRINT AT 10, 20, "OI"
Um ponto e vírgula entre dois itens imobiliza a posição de impressão, de forma que o segundo item siga logo após o primeiro.

Exemplo:
PRINT "A = ; 25
Uma vírgula, por outro lado, move a posição de impressão de, no mínimo, um lugar; e, após isso, tantos quantos forem necessários para deixá-la na coluna 0 ou 16, introduzindo uma nova linha, se necessário. Exemplo: PRINT "JOÃO", "MARIA"
Ao fim de uma instrução PRINT, se ela não terminar por ponto e vírgula ou vírgula, uma nova linha será colocada.
Erro 4 (fora da memória) pode ocorrer com 3k ou menos memória.
Erro 5 significa que a tela está repleta.
Em ambos os casos, a solução é digitar CONT, que limpará a tela e continuará.

REM...	Identifica comentários em um programa. (Abrev. REMark); ignora o restante da linha.
RUN	Executa um programa a partir da primeira linha.
RUN *n*	Executa um programa a partir da linha m. Exemplo: RUN 2Ø
STOP	Interrompe a execução do programa, mostrando a mensagem 9.
INT *n*	Função para obter a parte inteira de um número. Exemplo: PRINT INT 23.5 (sempre arredondando para menos).
LEN String	Comprimento da string.
LN Número	Logarítimo natural (na base e). Erro A se $x \leqslant 0$; 0 se $x \neq 0$; 1 se $x = 0$ NOT tem prioridade 4.
PI	Função que retorna o valor do número π (3,14159265...).
SGN *n*	Função que determina o sinal de um número. Nº Resultado positivo 1 zero 0 negativo -1 Exemplo: LET X = SGN (-25)
SQR *n*	Calcula a raiz quadrada do número n. Exemplo: PRINT SQR (24).

COMPUTADOR TAMBÉM SABE DECIDIR

Até agora, você aprendeu como programar o computador para:

Programa	Comando
entrada de dados:	INPUT
cáculos:	LET
resultados na tela:	PRINT

Estes comandos são muito úteis para fazer uma infinidade de programas. Com elas, podemos transformar o computador em uma sofisticada calculadora matemática, capaz de resolver problemas de cálculo de muitos tipos.

Entretanto, o computador pode fazer muito mais do que isso. Ele pode, por exemplo, aprender a tomar decisões sozinho!

O comando que permite fazer isto em BASIC se denomina IF. Em inglês, IF significa SE (ou seja, expressa uma determinada *condição*). Por exemplo:

```
300 IF X = 0 THEN PRINT
"O NUMERO E IGUAL A ZERO"
```

Este comando está dando a seguinte ordem para o computador:

SE o número X for igual a 0, ENTÃO imprima a mensagem.

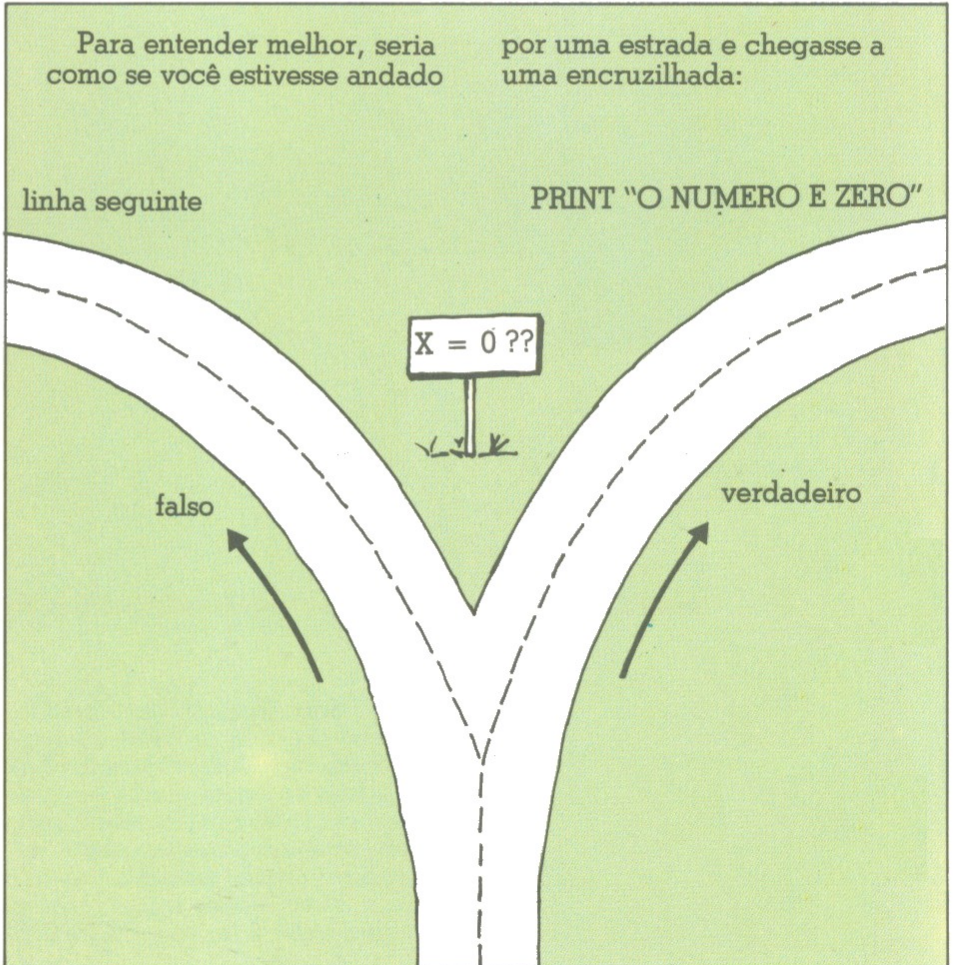
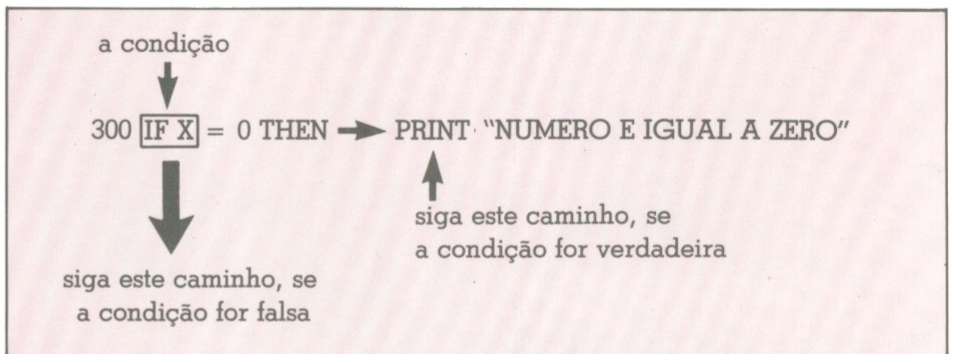
A palavra THEN significa ENTÃO, em inglês, e é sempre usada em conjunto com o IF.

O comando IF manda o computador tomar uma decisão muito simples. Ele ordena o computador a fazer uma determinada operação, se a condição testada for *verdadeira*.

No exemplo acima, se o número contido na variável X for igual a 0, a *condição testada* (X = 0) será verdadeira, e o computador mostrará na tela a mensagem:

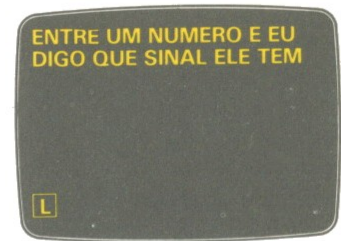
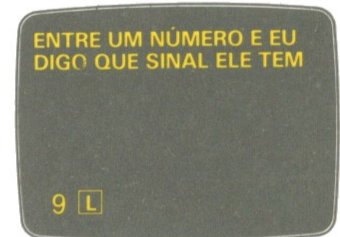
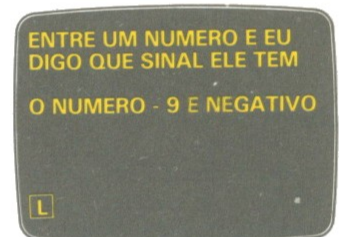
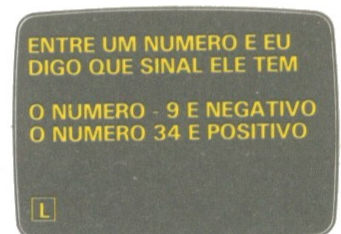
O NUMERO E IGUAL A ZERO

E se a condição não for verdadeira?

Então o computador não obedecerá a instrução PRINT na linha 300 e passará para a linha seguinte do progama:

Você tem dois caminhos para seguir, dependendo da decisão que tomar. Se o que estiver escrito na placa é verdadeiro, vá para a direita. Se for falso, vá para a esquerda!

Vamos ver agora um programa bem simples, para mostrar como funciona o comando IF numa situação real. Neste programa, a pessoa digita um número qualquer no teclado, e o computador informa se ele é zero, positivo ou negativo.

```
 10 REM — PROGRAMA PARA TESTAR SE UM NUMERO
 20 REM — E ZERO, POSITIVO OU NEGATIVO
 30 REM ______________
 40 REM
 50 REM — ENTRADA DE UM NUMERO
 60 REM
 70 PRINT "ENTRE UM NUMERO E EU DIGO QUE SINAL ELE"
 80 PRINT "TEM"
 90 INPUT X
100 REM
110 REM — TESTA O NUMERO
120 REM
125 PRINT
130 IF X = 0 THEN PRINT "O NUMERO E ZERO"
140 IF X > 0 THEN PRINT "O NUMERO "X;" POSITIVO"
150 IF X < 0 THEN PRINT "O NUMERO ";X;" E NEGATIVO"
160 PRINT
170 GOTO 90
```

Vamos ver como funciona este programa. Depois de digitar o comando RUN, aparece na tela

Então, se eu digitar, por exemplo, o número - 9:

E o computador responde:

Se eu digitar agora o número 34:

E assim por diante.

Para interromper o programa, podemos pressionar a tecla STOP.

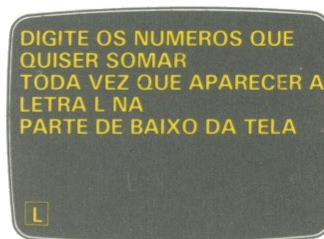
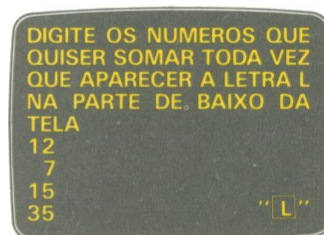
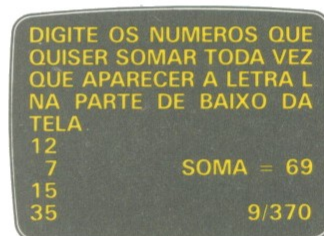
Outro uso bastante comum do IF é para assinalar quando uma determinada parte de um programa deve ser terminada e outra parte deve ser iniciada.

Veja o programa abaixo:

```
 10 REM — MAQUINA DE SOMAR
 20 REM __________
 30 REM
 40 REM — INSTRUCOES
 50 REM
 60 PRINT "DIGITE OS NUMEROS QUE QUISER SOMAR"
 70 PRINT "TODA VEZ QUE APARECER A LETRA L NA"
 80 PRINT "PARTE DE BAIXO DA TELA"
 90 PRINT
100 PRINT "PARA OBTER A SOMA, DIGITE 0"
110 PRINT
120 REM
130 REM — ZERA O SOMADOR
140 REM
150 LET SOMA = 0
160 REM
170 REM — ENTRADA DA PARCELA
180 REM
190 INPUT D
200 REM
210 REM — SE FOR ZERO, PULA PARA 350
220 REM
230 IF D = 0 THEN GOTO 350
240 REM
250 REM — SE NAO FOR, SOMA O DADO
260 REM
265 PRINT D
270 LET SOMA = SOMA + D
280 REM
290 REM — E VOLTA A 190 DE NOVO
300 REM
310 GOTO 190
320 REM
330 REM — FIM DA ENTRADA. MOSTRA SOMA
340 REM
350 PRINT
360 PRINT "SOMA = ";SOMA
370 STOP
```

O programa funciona como uma máquina de somar. O número de parcelas a serem somadas não é conhecido de antemão. Assim, a parte do programa entre as linhas 190 e 310 forma uma **alça de repetição**, que serve para pedir os valores para o usuário (através do comando INPUT, na linha 190), e somá-los usando o contador SOMA.

Para interromper esta alça, quando todos os números tiverem sido digitados, tenho que informar isso de algum modo ao computador. Poderíamos, neste caso, digitar o número zero. Um teste, dentro do programa, desvia o processamento para outra parte (linha 350 a 370), que vai mostrar o resultado.

Vamos ver um exemplo de utilização do programa:

Se eu digitar os números 12, 7, 15 e 35, um após o outro (pressionando a tecla NEW LINE ou ENTER, após cada um), teremos a seguinte tela:

Agora, se quisermos terminar esta parte do programa e ver o resultado da soma (linhas 350 a 370), vamos digitar 0 e ocorrerá o seguinte:

Veja quais seriam os números armazenados nas variáveis D e SOMA, no exemplo acima.

Número digitado	Variável D	Variável SOMA
12	12	12
7	7	19
15	15	34
35	35	69
0	0	69 e acaba...

Se eu não tivesse a linha de teste com o IF, seria impossível dizer ao computador quando ele deveria mostrar o resultado da SOMA.

Note como funciona a linha de teste

230 IF D = 0 THEN GOTO 350

Esta frase tem o seguinte significado em português:

Se o valor de D for igual a zero, ENTÃO vá para 350.

Quando o computador encontra esta linha de programa, o que fará ele?

1. Compara o número armazenado na variável D com zero;

2. se este número for igual a zero, então obedece o comando que está na mesma linha que o IF, logo depois do THEN, ou seja, GOTO 350;

3. se este número não for igual a zero, continua seguindo o programa, a partir da linha seguinte (ou seja, a linha 240).

Mais usos para o IF

O comando IF é interessante porque permite testar informações de muitas maneiras diferentes. Ele permite, por exemplo, verificar se a resposta a uma pergunta feita ao computador está correta ou não.

Vamos entender como funciona este programa, passo a passo:

```
10 REM ______ TESTE DE GEOGRAFIA
20 REM ____________________________
30 REM
40 REM ______ O COMPUTADOR FAZ UMA PERGUNTA
50 REM
60 PRINT "COMO SE CHAMA A CAPITAL DA ITALIA?"
70 INPUT RESPOSTA$
80 REM
90 REM ______ VAMOS VER SE A RESPOSTA
   ESTA CERTA
100 REM
110 IF RESPOSTA$="ROMA" THEN GOTO 200
120 REM
130 REM ______ NAO, ERROU
140 REM
150 CLS
160 PRINT "ERROU. A CAPITAL DA ITALIA E ROMA"
170 STOP
180 REM
185 REM ______ PULA PRA CA, SE A RESPOSTA
   FOI CORRETA
190 REM
200 CLS
210 PRINT "ACERTOU, VOCE SABE GEOGRAFIA, HEIN?"
220 STOP
```

— linha 60: o programa mostra a pergunta, no alto da tela

— linha 70: fica esperando a resposta digitada no teclado, e a guarda na variável chamada RESPOSTA$

— linha 110: compara o que está em RESPOSTA$ com a constante ROMA.

Se for verdadeiro, ou seja, se a pessoa digitou ROMA, ENTÃO pula para a linha n. 200, para dar os parabéns.

— linha 150 a 160: limpa a tela (CLS), diz que errou, e mostra a resposta correta. Depois pára, interrompe o programa.

— linha 200 a 220: limpa a tela, e dá os parabéns. Depois interrompe o programa.

Como funciona o comando IF?

Com os exemplos acima, já podemos deduzir alguns princípios sobre o funcionamento do comando IF:

— uma linha com o comando IF é composta de quatro partes:

1. a palavra IF

2. uma comparação entre duas coisas

3. a palavra THEN

4. uma linha com um comando que deve ser executado, se o resultado da comparação for VERDADEIRO

As palavras IF e THEN são fixas, ou seja, sempre devem constar na linha.

Vamos ver o que o BASIC permite que coloquemos nos itens 2 e 4.

O item 2 é chamado de CONDIÇÃO, ou ainda de CLÁUSULA LÓGICA.

O item 4 é chamado de AÇÃO ou CONSEQÜÊNCIA.

A condição

Em geral, a cláusula lógica é escrita de tal forma que permita uma comparação.

Por exemplo:

IF X = 5 THEN ...

testa se o valor da variável numérica X é *igual* a 5

IF A > B THEN ...

testa se o valor da variável numérica A é *maior* do que o valor da variável numérica B

IF 13 + (25 - 3) / 2 < C THEN ...

testa se a expressão aritmética à esquerda é *menor do que* o valor da variável numérica C

IF R$ = "SIM" THEN ...

testa se a palavra armazenada na variável alfanumérica R$ *é igual* à constante alfanumérica SIM

Só existem dois resultados possíveis para uma expressão lógica: verdadeiro ou falso.

Se a EXPRESSÃO resulta VERDADEIRA, a AÇÃO é executada.

Se a EXPRESSÃO resulta FALSA, a AÇÃO não é executada.

Os sinais de igual (=), maior (>) e menor (<) são chamados de operações relacionais, ou seja, permitem formar uma expressão que contenha uma igualdade ou uma desigualdade (como na matemática).

As seguintes operações relacionais são permitidas em BASIC:

=	igual
<	menor que
>	maior que
< =	menor ou igual a
> =	maior ou igual a
< >	diferente de (não é igual)

Para digitar estas operações no teclado, note que existe algumas teclas rotuladas com estes sinais:

tecla L	=
tecla N	<
tecla M	>
tecla R	< =
tecla Y	> =
tecla T	< >

Para entrar a operação, mantenha pressionada a tecla SHIFT, enquanto pressiona a tecla correspondente ao sinal desejado.

Preste atenção em uma coisa: vai haver erro se você tentar obter o sinal > = pressionando a tecla > e depois a tecla =.

Este sinal só pode ser obtido corretamente por meio de sua tecla especial, e assim todos os outros.

Uma comparação lógica deve sempre ter variáveis ou constantes do mesmo tipo, na mesma expressão. Caso contrário, ocorrerá erro.

Por exemplo:

IF COMIDA = "BATATA" THEN ...

não está correto, porque estamos tentando comparar uma variável numérica com uma constante alfabética.

Ou então:

IF X $ > 24 THEN ...

também não dará certo, porque estamos colocando na mesma expressão lógica uma variável *alfanumérica* e uma constante *numérica*.

A ação

O que vem na mesma linha do IF, logo depois da expressão lógica, é a ação ou conseqüência a ser seguida, se ela for verdadeira.

No BASIC do TK-85, bem como no de todos os outros microcomputadores compatíveis com a linha Sinclair, a ação pode ser qualquer comando válido em BASIC:

Você já conhece alguns, que podem ser colocados sem problemas depois de um comando IF:

RUN
GOTO
CLS
PRINT
INPUT
STOP
LET

são os mais usados.

Eis aqui alguns exemplos:

IF SEXO$ = "FEMININO" THEN GOTO 300

IF T 20 > THEN STOP

IF S < 0 THEN PRINT "O NUMERO E NEGATIVO"

IF R <> 2 THEN INPUT P$

IF ANOS = "BISSEXTO" THEN LET DIAS = 366

IF LINHAS > 20 THEN CLS

IF FIM = 0 THEN RUN

Vamos ver mais alguns exemplos do uso do comando IF:

EXEMPLO 1: um programa para xingar uma pessoa!

```
10 REM - PROGRAMA PARA XINGAR UMA PESSOA
20 REM ______________________________________________
30 REM
40 REM __________ PERGUNTA O NOME E O SEXO DA PESSOA
50 REM
60 PRINT "COMO VOCE SE CHAMA?"
70 INPUT  NOME$
80 PRINT "QUAL E O SEU SEXO (MASCULINO OU FEMININO)?"
90 INPUT SEXO$
100 REM
110 REM __________________________ TESTA QUAL E O SEXO
120 REM
130 IF SEXO$ = "MASCULINO"
THEN LET XING$ = "BOBO"
140 IF SEXO$ = "FEMININO"
THEN LET XING$ = "BOBA"
150 REM
160 REM __________ ESCREVE O XINGAMENTO APROPRIADO
170 REM
180 PRINT
190 PRINT NOME$; "  VOCE E  ";XING$
200 STOP
```

Como vai ser o resultado deste programa?

O que acontecerá se a pessoa digitar de uma forma errada o sexo, ou seja, se não for exatamente igual a MASCULINO ou FEMININO?

Exemplo 2: programa para cálculo de raiz quadrada.

Vamos supor que queremos fazer um programa para calcular a raiz quadrada de um número qualquer. Queremos que o programa funcione assim: primeiro, a pessoa digita no teclado o número que desejar. Depois, o programa calcula a raiz quadrada e mostra o resultado na tela. Em seguida, pede outro número à pessoa, e assim por diante.

```
10 REM - PROGRAMA PARA CALCULO DE RAIZ QUADRADA
20 REM - PARA O COMPUTADOR TK-85
30 REM ______________________________________________
40 REM
50 REM ______________________ ENTRADA DE UM NUMERO
60 REM
70 PRINT "CALCULO DA RAIZ QUADRADA"
80 PRINT"= = = = = = = = = = = = = = = = = = = = = = =  ="
90 PRINT "DADO";
100 INPUT D
110 PRINT D;
115 REM
120 REM — CALCULO E IMPRESSAO
130 REM
140 LET R = SQR (D)
150 PRINT "RAIZ QUADRADA = ";R
160 REM
170 REM — VOLTA PARA O COMECO, PARA
    OUTRO NUMERO
180 REM
190 GOTO 90
```

É muito fácil entender como funciona o programa:
— nas linhas 90 E 100, o programa solicita o número a ser calculado;
— na linha 110, o número digitado é impresso na mesma linha;
— nas linhas 140 e 150, o programa calcula e mostra a raiz quadrada;
— na linha 190, o programa é desviado para o comando INPUT, de modo a possibilitar novo cálculo.

O GOTO 90 na linha 190 cria uma *"alça sem fim"*, ou seja, o programa sempre vai se repetir, infinitamente. Quando você quiser parar de utilizar o programa, basta pressionar a tecla STOP, com o programa na linha 90, ou pressionar a tecla BREAK, quando ele estiver calculando. Isto você já aprendeu.

O programa ficaria muito melhor, entretanto, se não fosse necessário pressionar estas teclas para interrompê-lo. Poderíamos fazer com que o computador "soubesse" quando deveria parar.

Uma sugestão interessante seria esta: se a pessoa entrar um valor determinado (por exemplo, zero), o programa parará automaticamente.

Para isto, precisamos testar se o valor armazenado na variável de entrada D é igual a zero, certo?

```
105 IF D = 0 THEN STOP
```

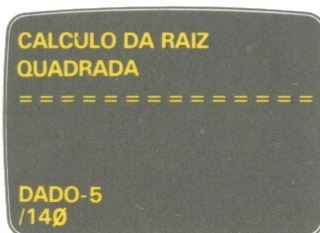
Agora, nosso programa ficará assim:

```
10 REM - PROGRAMA PARA CALCULAR RAIZ QUADRADA
20 REM - PARA O COMPUTADOR TK - 85
30 REM ____________________________________________
40 REM
50 REM — ENTRADA DE UM NUMERO
60 REM
70 PRINT "CALCULO DA RAIZ QUADRADA"
80 PRINT " = = = = = = = = = = = = = = = = = = = = = = = = "
90 PRINT "DADO";
100 INPUT D
101 REM
102 REM — TESTA SE O VALOR E ZERO
103 REM
105 IF D = 0 THEN STOP
115 REM
120 REM — CALCULO E IMPRESSAO
130 REM
140 LET R = SQR (D)
150 PRINT "RAIZ QUADRADA = "; R
160 REM
170 REM — VOLTA PARA O COMECO, PARA
    OUTRO NUMERO
180 REM
```

Agora parece que o programa já está bom, mas existe ainda outro probleminha a ser resolvido.

Não existe a raiz quadrada de um número negativo.

Assim, se você digitar o número -5, por exemplo, o programa será interrompido, e aparecerá a seguinte mensagem de erro:

```
CALCULO DA RAIZ
QUADRADA
= = = = = = = = = = = = = =

DADO-5
/14Ø
```

Isto quer dizer que a função na linha indicada (140) foi usada com um *argumento* (isto é, dado de entrada) indevido ou errado (não permitido ou impossível de ser calculado).

Não é conveniente deixar ocorrer este tipo de interrupção em um programa, porque às vezes a pessoa que está operando não sabe o que significa aquela mensagem de erro nem como voltar a utilizar o programa.

O ideal é que o *próprio programa se encarregue de verificar se o número digitado pode ou não ser usado.*

Para que façamos isto, é necessário realizar um teste. No nosso exemplo, teríamos que testar, *dentro do programa*, se o número digitado pela pessoa é válido ou não.

Se o número fosse incorreto, o programa poderia mostrar uma mensagem de advertência (por exemplo: NÃO PODE SER UM NÚMERO NEGATIVO, ou outra qualquer).

Por exemplo, poderíamos colocar em nosso programa a seguinte linha, logo após aquela que serve para entrar o número a ser calculado (linha 100, com o comando INPUT):

```
109 IF D < 0 THEN GOTO 250
```

Esta linha permitirá interceptar todos os números negativos. Se a cláusula

```
D < 0
```

for verdadeira, o programa pulará para a linha 250, onde será mostrada uma mensagem de advertência, para indicar o erro cometido.

Se ela for falsa (ou seja, o número é maior do que zero), o programa pulará para a linha seguinte (110), e o restante do cálculo será realizado.

Agora, com todas as modificações, nosso programa ficará assim:

```
10 REM - PROGRAMA PARA CALCULAR RAIZ QUADRADA
20 REM - PARA O COMPUTADOR TK - 85
30 REM ______________________________________________
40 REM
50 REM — ENTRADA DE UM NUMERO
60 REM
70 PRINT "CALCULO DA RAIZ QUADRADA"
80 PRINT " = = = = = = = = = = = = = = = = = = = = = = = "
90 PRINT "DADO";
100 INPUT D
101 REM
102 REM — TESTA SE O VALOR E ZERO
103 REM
105 IF D = 0 THEN STOP
106 REM
107 REM — TESTA SE O VALOR E NEGATIVO
108 REM
109 IF D<0 THEN GOTO 250
110 PRINT D;
115 REM
120 REM — CALCULO E IMPRESSAO
130 REM
140 LET R = SQR (D)
150 PRINT "RAIZ QUADRADA = "; R
160 REM
170 REM — VOLTA PARA O COMECO, PARA OUTRO NUMERO
180 REM
190 GOTO 90
200 REM
210 REM — MENSAGEM DE ERRO
220 REM
250 PRINT "NAO PODE SER UM NUMERO NEGATIVO"
260 GOTO 90
```

Coloque este programa no computador e teste com alguns números positivos e negativos para ver se ele funciona.

EXERCÍCIOS

1. Responda que valor o programa vai mostrar na tela (linha 3Ø) ao ser executado.

a - 1Ø LET C = 34
  2Ø LET D = 25
  3Ø IF C > D THEN PRINT "NUMEROS DIFERENTES"

b - 1Ø LET A = 234.5
  2Ø LET B = 234
  3Ø IF B-A < Ø THEN PRINT "RAIZ QUADRADA IMPOSSIVEL"
  4Ø IF B-A > Ø THEN PRINT SQR(B-A)

c - 1Ø LET A$ = "BANANA"
  2Ø LET B$ = "AVESTRUZ"
  3Ø IF A$ < > B$ THEN PRINT "SAO DUAS PALAVRAS DIFERENTES"

2. Escreva o significado das operações relacionais:

< > ____________
< = ____________
< ____________
> ____________
= ____________
> = ____________

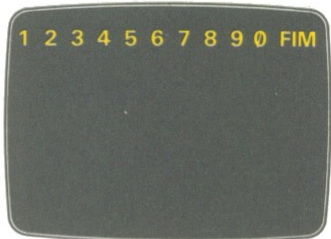
3. Faça um programa para calcular o produto de dois ou mais números.
Use o exemplo da "máquina de somar" como modelo.

4. Prolongue o programa do teste de geografia, de modo a incluir outras questões (por exemplo, sobre a capital da Inglaterra, da França e da Espanha).

5. Escreva por extenso o que significa cada teste IF abaixo:

a - IF A = 12.5 THEN...
b - IF SQR(MEDIA) > 12 THEN...
c - IF NOME$ = "DELFIM NETO" THEN...
d - IF B**2 > INT(B) THEN...

6. Escreva ao lado de cada comando IF qual é o erro:

a - IF MEDIDA = "COLHER" THEN RUN
b - IF X = < B$ THEN PRINT "SAO DIFERENTES"
c - IF B > 12 PRINT PRINT
d - IF LADO$ = 25 THEN STOP
e - STOP IF LADO = 25
f - IF B2 + E5 + "F4" = C22 THEN CLS

7. Faça um programa para testar se o resultado da divisão de um número por outro tem resto maior do que zero.

Dica:
```
10 INPUT N1
20 INPUT N2
30 LET D=N1/N2
40 IF...
```
Use a função INT (inteiro de) para verificar.

8. Explique o que faz o programa abaixo com os valores de entrada (A e B) indicados.

```
10 INPUT A
20 INPUT B
30 LET N=0
40 LET N=N+1
50 IF N=B THEN STOP
60 LET A=A+2
70 PRINT A; " ";
80 GOTO 40
```

Valores: a - A=2, B=5
b - A=4, B=1
c - A=0, B=10
d - A=1, B=.1

9. Estude o programa abaixo e escreva as mensagens que serão escritas na tela quando o programa for executado:

```
10 PRINT "CURSO DE PROGRAMAÇÃO BASIC"
20 GOTO 75
30 PRINT "EDITORA ABRIL CULTURAL"
40 PRINT
50 GOTO 140
60 PRINT "BASIC E MUITO FACIL"
70 GOTO 90
75 PRINT "APRENDER";
80 GOTO 60
90 LET A=8
100 IF A=8 THEN 110
105 PRINT "BASIC E MUITO DIFICIL"
110 STOP
140 PRINT "FIM DA MENSAGEM"
```

Responda também: quais são as linhas do programa que nunca serão executadas. Por quê?

10. Faça um programa que imprima na tela os números de 10 a 20.

Mais repetições

Como vimos na primeira parte deste capítulo, um dos usos principais para o comando IF é observado na contagem e repetição de certos segmentos de um programa (ou seja, o que chamamos de *alças de repetição*).

Para isto, ele é empregado em conjunto com uma variável chamada *acumuladora* ou *contadora*, que serve para registrar quantas vezes cada coisa foi repetida.

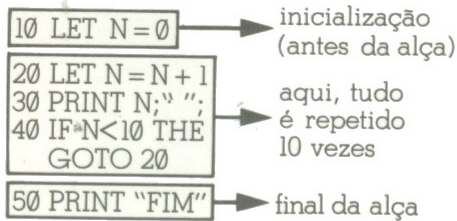
Por exemplo: um programa para contar de 1 a 10, mostrando na tela o seguinte resultado:

poderia ser programado da seguinte maneira:

```
10 LET N=0
20 LET N=N+1
30 PRINT N;" ";
40 IF N<10 THEN GOTO 20
50 PRINT "FIM"
```

Vamos entender melhor como funciona este programa:

10 LET N=0	N será a variável contadora. Por isso, devemos inicializá-la (no caso, igualá-la a zero). Isto deve ser feito *antes* de se começar a alça de repetição.
20 LET N=N+1	Aqui é o começo da alça de repetição. Somamos 1 ao valor anterior armazenado em N (isto é, incrementamos ou acumulamos o valor da contadora N).
30 PRINT N;" ";	Aqui imprimimos o valor atualizado de N, que será maior cada vez que a alça for repetida.
40 IF N<10 THEN GOTO 20	Este é o fim da alça de repetição. Testamos se o valor incrementado na variável N já atingiu 10. Se ainda não tiver atingido, volta-se para a linha 20, para nova repetição.
50 PRINT "FIM"	Se o teste IF da linha anterior não for verdadeiro, ou seja, se N for igual ou maior do que 10, então o programa sai da alça e encerra o processamento, imprimindo FIM.

Portanto, vemos que este programa tem três partes:

se quisermos, poderemos acrescentar outros comandos, fora da alça.

Note que este programa poderia ser escrito de muitas outras maneiras, com exatamente o mesmo resultado, em termos de processamento:

A)
```
10 LET N=0
20 IF N>10 THEN 60
30 LET N=N+1
40 PRINT N;" ";
50 GOTO 20
60 PRINT "FIM"
```

B)
```
10 LET N=1
20 PRINT N;" ";
30 LET N=N+1
40 IF N<=10 THEN 20
50 PRINT "FIM"
```

C)
```
10 LET N=1
20 IF N>10 THEN 50
30 PRINT N;" ";
35 LET N=N+1
40 GOTO 20
50 PRINT "FIM"
```

Na variante A, o teste é feito antes do incremento. A variável contadora é igualada a zero na inicialização, por isso, a impressão (PRINT) tem que ser feita depois do incremento (N=N+1).

Na variante B, a contadora é igualada a 1. Por isso, primeiro devemos imprimir seu conteúdo para somente depois incrementá-lo. Neste exemplo, o teste é feito no final da alça.

Na variante C, temos uma mistura de A e B: o teste é feito no início da alça, mas a contadora é igualada a 1 também no início.

Como essas, você poderia imaginar ainda um grande número de combinações das linhas, obtendo o mesmo resultado. Por isso, quase nunca existe uma única solução para um programa: várias soluções podem levar ao mesmo resultado.

Mas cuidado: um pequeno erro no operador relacional, ou na ordem de colocação das linhas, pode levar a um resultado diferente do esperado. Por exemplo:

```
10 PRINT N=0
20 IF N > 10 THEN GOTO 60
30 PRINT N; " ";
40 LET N=N+1
50 GOTO 20
60 PRINT "FIM"
```

imprimirá 10 números, conforme o previsto, porém o fará de 0 a 9. Experimente!

Que linha você mudaria para obter um resultado correto?

Modificando ligeiramente a lógica do programa acima, podemos fazê-lo contar de 2 em 2, de 3 em 3 etc.

Por exemplo, queremos fazer com que o programa imprima todos os números pares de 2 a 10:

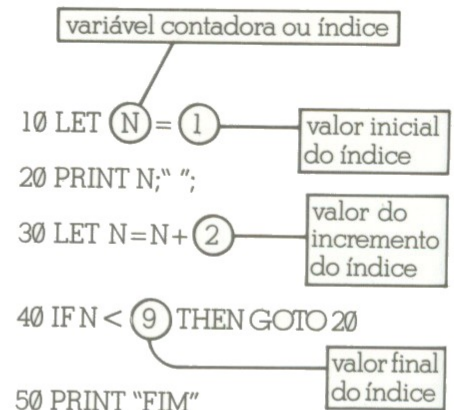
```
10 LET N=0
20 LET N=N+2
30 PRINT N; " ";
40 IF N<10 THEN GOTO 20
50 PRINT "FIM"
```

Na tela, será apresentado o seguinte resultado:

Para obter os números ímpares de 1 a 9, bastariam estas mudanças:

```
10 LET N=1
20 PRINT N;" ";
30 LET N=N+2
40 IF N<9 THEN GOTO 20
50 PRINT "FIM"
```

Obteríamos o seguinte resultado:

Portanto, podemos ver claramente que todo programa de *repetição controlada* de alças (isto é, através de um certo número fixo de repetições), tem sempre os mesmos elementos básicos:

A repetição (da linha 20 à 40) se dá entre os valores inicial e final fixados no programa para a variável contadora, ou índice.
No programa acima, as linhas 10, 30 e 40 servem exclusivamente para montar a lógica de repetição:

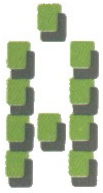
linha 10 - inicialização da contadora
linha 30 - incremento da contadora
linha 40 - testa se o conteúdo da contadora atingiu o valor final.

Portanto, na realidade, apenas a linha 20 é o "miolo" da alça.

Podemos colocar quantos comandos quisermos entre os comandos de início e fim de uma alça. Neste caso, todos serão repetidos tantas vezes quantas determinar a contagem.

Por exemplo, vamos fazer um programa para imprimir na tela o quadrado e o cubo dos múltiplos de 2 (2, 4, 6, 8 etc.), até 20.

```
10 LET N=2
20 IF N>20 THEN 80
30 LET Q=N*N
40 LET C=N*N*N
50 PRINT N, Q, C
60 LET N=N+2
70 GOTO 20
80 STOP
```

Observe que neste exemplo as linhas de 30 a 50 constituem os comandos a serem repetidos pela alça. As linhas 10, 20, 60 e 70 são usadas para incrementar e testar esta alça.

Alças variáveis

Outra possibilidade muito interessante é substituir as constantes correspondentes aos valores inicial, final e de incremento de uma alça por variáveis numéricas. Isto permite que, com um único programa, possamos fazer uma infinidade de repetições diferentes.

Por exemplo, vamos fazer um programa que saiba contar de 1 até o número que quisermos (que será fornecido pelo teclado):

```
 10 REM --- PROGRAMA QUE SABE CONTAR
 20 REM --------------------------------
 30 REM
 40 REM --- MENSAGEM INICIAL
 50 REM
 60 PRINT "EU SEI CONTAR DE 1 ATE O NUMERO"
 70 PRINT "QUE VOCE QUISER."
 80 PRINT
 90 PRINT "DIGITE O NUMERO QUE QUISER = ";
100 INPUT NUM
110 PRINT NUM
120 PRINT
130 REM
140 REM --- INICIALIZACAO DA CONTADORA
150 REM
160 LET CONT=Ø
170 REM
180 REM --- ALCA DE REPETICAO
190 REM
200 LET CONT=CONT+1
210 PRINT CONT
220 IF CONT<NUM THEN GOTO 200
230 STOP
```

Da mesma forma, o incremento também poderia ser colocado como variável.

Por exemplo, este programa mostra na tela a tabuada de multiplicar que quisermos.

```
10 REM --- TABUADA DE
   MULTIPLICAR
20 REM ------------------------------
30 REM
40 PRINT "QUE TABUADA
   VOCE QUER ESTUDAR?"
50 INPUT T
60 CLS
70 PRINT "EIS A TABUADA DE ";T
80 PRINT
90 LET N=T
100 PRINT N;" "; N*T
110 LET N=N+T
120 IF N<T*10 THEN GOTO 100
130 STOP
```

Um exemplo de execução deste programa seria o seguinte:

EIS A TABUADA DE 4:

4 8 12 16 20 24 28 32 40

9/ 130

Os valores de incício, fim e incremento da alça não precisam ser inteiros ou crescentes, como nos exemplos até agora.

Se o valor do incremento for negativo (neste caso, o nome apropriado seria *decremento*), a contagem será retroativa, ou seja, o valor final deverá ser menor do que o inicial.

Por exemplo, vamos fazer um programa que faça uma contagem retroativa para simular o lançamento de um foguete!

O programa pode ser bem simples:

```
10 REM --- PROGRAMA PARA CON-
   TAGEM REGRESSIVA
20 REM -----------------------------------
30 REM
40 PRINT "VAMOS LANCAR UM FO-
   GUETE DA"
50 PRINT "BARREIRA DO INFER-
   NO..."
60 PRINT
70 PRINT "QUANTOS SEGUNDOS
   FALTAM?"
80 INPUT SEG
90 REM
100 REM      CONTAGEM REGRES-
    SIVA
110 REM
120 CLS
130 PRINT "ATENCAO"
140 PRINT
150 PRINT SEG;" ";
160 LET SEG=SEG-1
170 IF SEG>Ø THEN 150
180 PRINT
190 PRINT
200 PRINT "FOGO... "
210 STOP
```

Observe que neste programa a variável contadora (SEG) é inicializada pelo próprio comando INPUT, na linha 80.

Com o programa seguinte, você pode ver também que os valores da alça não precisam ser inteiros (ou seja, podem ser fracionários):

```
10 PRINT "VALOR INICIAL DO INDI-
   CE?"
20 INPUT I
30 PRINT "VALOR DO INCREMENTO
   DO INDICE?"
40 INPUT S
50 PRINT "VALOR FINAL DO INDI-
   CE?"
```

```
60 INPUT F
70 LET N=I
80 PRINT N;" ";
90 LET N=N+S
100 IF N F THEN GOTO 80
110 PRINT
120 STOP
```

Por exemplo, se colocarmos os valores I=4, S=0.5 e F=6, teremos o resultado:

Tem um jeito mais fácil

Como a programação de alças de repetição é um recurso muito utilizado, a linguagem BASIC foi dotada de um comando que permite fazê-la de modo mais fácil do que usando o comando IF.

Este comando é chamado de FOR que, em inglês, significa PARA.

Para construir uma alça de repetição com o comando FOR, é necessário marcar também o final da mesma por meio de segundo comando, chamado NEXT (SEGUINTE, em português), e que sempre é usado em conjunto com o FOR.

Siga o exemplo abaixo, para compreender de que jeito os comandos FOR... NEXT são utilizados para obter uma repetição.

USANDO IF

```
10 LET N=1
20 PRINT N
30 LET N=N+1
40 IF N <10 THEN GOTO 20
50 PRINT "FIM"
```

USANDO FOR ... NEXT

```
10 FOR N=1 TO 10
20 PRINT N
30 NEXT N

50 PRINT "FIM"
```

O comando na linha 10 do programa à direita está dizendo para o computador:

10 repita PARA a variável N, indo de 1 ATÉ 10, de 1 em 1.

Todos os comandos que estiverem entre a linha 10 e o próximo NEXT serão repetidos tantas vezes quantas o número indicar (no exemplo, apenas a linha 20).

O comando na linha 30 está dizendo para o computador:

30 PRÓXIMA repetição com a variável N.

O que significa que o programa retorna à linha 20 (ou a qualquer linha que estiver imediatamente em seguida à linha com o comando FOR), para nova repetição, *desde que o valor da contadora não tenha atingido o seu final*.

Portanto, o comando FOR serve para fixar:

1. o nome da variável de contagem (índice)
2. o valor inicial do índice
3. o valor final do índice
4. onde deve começar a alça de repetição

Enquanto o comando NEXT serve para realizar o seguinte:

1. incrementar o valor do índice
2. testar se ele atingiu o valor final
3. indicar o final da alça

Depois da alça FOR...NEXT ter sido repetida *n* vezes, o programa retoma o seu fluxo normal, a partir da linha imediatamente após a linha que contém o NEXT.

Atenção: na linguagem BASIC para microcomputadores compatíveis com a lógica Sinclair (marcas TK 82C, TK 83, TK 85, CP 200 etc.), *é obrigatória a colocação do nome da variável-índice no comando NEXT*. Em outros computadores, nem sempre isto é necessário.

Você deve ter notado que o comando FOR, como foi mostrado até agora, não especificou o valor do incremento (quantidade a ser adicionada ao índice, a partir do valor inicial).

Se nada for especificado no comando FOR, o valor do incremento é igual a 1.

Se quisermos colocar um incremento diferente de 1, usamos o comando STEP (que significa PASSO), ao final do comando FOR:

```
10 FOR I=2 TO 10 STEP 2
20 PRINT I;" ";
30 NEXT I
40 PRINT "FIM"
```

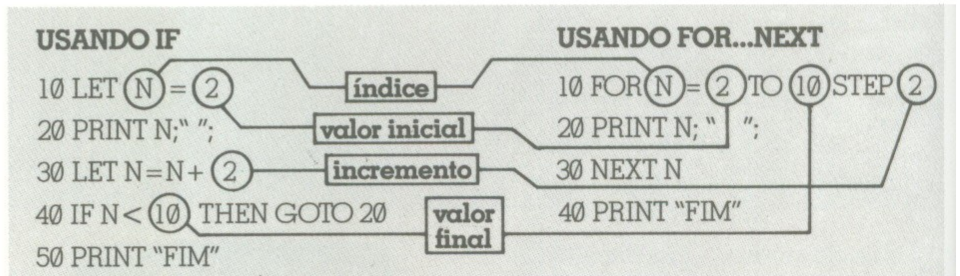
Este programa vai mostrar todos os números pares de 2 a 10. O incremento necessário (no caso, 2) é indicado pela palavra STEP.

Portanto:

FOR N=1 TO 10 STEP 1

é a mesma coisa que

FOR N=1 TO 10

Portanto, se compararmos as duas formas de programar uma alça, poderemos facilmente identificar os seus elementos básicos:

Os comandos que estão entre um FOR e seu NEXT correspondente são agrupados no chamado *alcance* da alça. Um outro nome que você poderá encontrar para alça, em outros livros e revistas, é *malha*.

Ao contrário das alças construídas com o comando IF, as alças com o comando FOR...NEXT admitem apenas uma estruturação.

Note como o programa para elaboração da tabela de quadrados e de cubos fica muito mais simples:

```
10 FOR N=2 TO 20 STEP 2
20 LET Q=N*N
30 LET C=N*N*N
40 PRINT N,Q,C
50 NEXT N
60 STOP
```

Várias linhas de programação e uma complicação desnecessária são evitadas.

As linhas 20, 30 e 40 são o alcance da alça, que vai de 20 a 50.

Note também que o comando FOR aceita também especificações fracionárias ou negativas

Por exemplo, o nosso programinha de contagem regressiva para os foguetes ficaria assim:

```
10 REM --- PROGRAMA PARA
   CONTAGEM REGRESSIVA
20 REM ------------------------------------
30 REM
40 PRINT "VAMOS LANCAR UM
   FOGUETE DA"
50 PRINT "BARREIRA DO INFER-
   NO..."
60 PRINT
70 PRINT "QUANTOS SEGUNDOS
   FALTAM?"
80 INPUT SEG
90 REM
100 REM --- CONTAGEM
    REGRESSIVA
110 REM
120 CLS
130 PRINT "ATENCAO"
140 PRINT
150 FOR S=SEG TO 1 STEP -1
160 PRINT S;" ";
170 NEXT S
180 PRINT
190 PRINT
200 PRINT "FOGO..."
210 STOP
```

Vale a pena anotar algumas diferenças importantes entre este programa e o que mostramos anteriormente, usando o comando IF:

1. a variável SEG (valor inicial da contagem regressiva) não pode ser usada também como variável de índice ou contadora. Por isso usamos o valor S;

2. o valor inicial (SEG) é menor do que o final (1);

3. apesar do incremento ser 1, em valor absoluto, como é negativo, precisa ser especificado pelo comando STEP;

4. todas as especificações do comando FOR podem ser constantes ou variáveis, ou ainda expressões aritméticas válidas, indistintamente.

   Veja este exemplo:

```
10 PRINT "QUE TABUADA VOCE
   QUER VER?"
20 INPUT T
30 PRINT "A PARTIR DE QUE VA-
   LOR?"
40 INPUT I
50 PRINT "ATÉ QUE VALOR?"
60 INPUT F
70 CLS
80 PRINT "EIS A TABUADA DOS ";T
90 PRINT
100 FOR N=I TO F STEP T
110 PRINT N;" ";
120 NEXT N
130 STOP
```

Alguns errinhos que você deve evitar, quando for usar os comandos FOR... NEXT:

1. na linha com o comando FOR, não podem ser omitidos o sinal de igualdade (=), o valor inicial (constante, variável ou expressão), a palavra TO, o valor final (constante, variável ou expressão), *nesta ordem*;

2. a palavra STEP é opcional, mas, se colocada, deve ser obrigatoriamente seguida do valor do mesmo (também uma constante, variável ou expressão válidas em BASIC);

3. não pode existir um comando FOR sem o NEXT correspondente nem um NEXT sem FOR;

4. é obrigatório colocar o nome da variável de índice no comando NEXT;

5. o comando FOR deve ser encontrado pelo programa antes do comando NEXT que lhe corresponder.

Verifique, na tabela de mensagens de erros (página 33), quais são os códigos que ocorrem quando cada uma das regras acima for desrespeitada na programação.

esperando o tempo passar...

Uma das utilidades das alças de repetição é fazer o tempo passar. Em outras palavras, se quisermos fazer com que o computador aguarde um certo período de tempo, entre uma fase e outra de um programa, podemos recorrer a uma alça FOX NEXT com um grande número de repetições, mas sem nenhum comando de entrada ou saída a seu alcance. Quanto maior for o número de repetições, maior será o tempo obtido.

Por exemplo:

```
10 PRINT "COMPUTADOR";
20 FOR I=1 TO 500
30 NEXT I
40 GOTO 10
```

Este programa escreve a palavra COMPUTADOR várias vezes, uma após a outra, até encher toda a tela.

Entretanto, a "alça vazia" (isto é, que não contém nenhum comando) das linhas 20 e 30 cria um aumento de tempo que, no TK 85 e em outros microcomputadores do mesmo tipo, corresponde a cerca de 2 segundos entre uma palavra e outra.

Outro exemplo, para fazer uma palavra piscar na tela:

```
10 PRINT "COMPUTADOR"
20 FOR I=1 TO 100
30 NEXT I
40 CLS
50 PRINT "           "
60 FOR I=1 TO 100
70 NEXT I
80 GOTO 10
```

Neste programa, a linha 10 escreve a palavra COMPUTADOR na primeira linha da tela. Em seguida, há um pequeno aumento de tempo, dado pela alça nas linhas 20 e 30, que faz com que esta palavra permaneça na tela por um certo tempo. Depois, a linha 40 apaga toda a tela. As linhas 60 e 70 criam um novo au-

mento de tempo, que a mantém apagada por igual período, e a linha 80 realiza a repetição do processo.

Muitos computadores, como é o caso dos compatíveis com a lógica Sinclair, têm um comando especial para obter aumentos de tempo, ou pausas, como são tecnicamente chamadas.

Este comando é denominado PAUSE. Procure-o no teclado de seu computador.

O programa acima poderia ser escrito de forma muito mais simples, usando-se o comando PAUSE:

```
10 PRINT "COMPUTADOR"
20 PAUSE 20
30 CLS
40 PAUSE 20
50 GOTO 10
```

Quando o computador encontra o comando PAUSE, ele interrompe todo o processamento e continua a mostrar tudo o que foi escrito na tela durante *n* quadros (*n* é o número, maior que zero e menor que 32767, que indica ao computador quantos quadros ele deve esperar). Normalmente, a tela é renovada, em um aparelho de TV, à velocidade de 60 telas ou quadros por segundo. Isto nos dá a impressão de que ela é fixa, mas na realidade ele está piscando tão depressa, que nem percebemos.

Assim, no programa acima, o número 20, depois do comando PAUSE, faz o computador esperar cerca de 20/60 de segundo, ou 1/3 de segundo, para depois prosseguir.

Quais as vantagens ou desvantagens da alça FOR NEXT em relação ao comando PAUSE?

Bem, em primeiro lugar, a pausa dada por FOR NEXT funciona para qualquer computador, em qualquer "dialeto" (variação) da linguagem BASIC. Por isso, se você quiser fazer um programa que possa ser colocado em vários tipos de computador, use FOR... NEXT.

Em segundo lugar, o comando PAUSE tem uma peculiaridade: nos computadores tipo TK, a cada 60 quadros (aproximadamente 1 segundo), a tela de vídeo pisca perceptivelmente, produzindo uma imagem pouco estável. Este efeito artificial não ocorre com o FOR... NEXT.

Por outro lado, uma vantagem nítida que o PAUSE tem sobre o FOR... NEXT, é que utiliza um número menor de linhas do que este (portanto, o programa gasta menos memória, é mais curto e mais fácil de ser entendido).

Vamos ver mais algumas aplicações interessantes para o PAUSE.

No programa de contagem regressiva anterior, podemos tornar mais real a simulação de uma contagem se fizermos assim:

```
10 REM --- PROGRAMA PARA CON-
     TAGEM REGRESSIVA
20 REM ------------------------------------
30 REM
40 PRINT "VAMOS LANCAR UM
     FOGUETE DA"
50 PRINT "BARREIRA DO INFER-
     NO..."
60 PRINT
70 PRINT "QUANTOS SEGUNDOS
     FALTAM?"
80 INPUT SEG
90 REM
100 REM --- CONTAGEM
      REGRESSIVA
110 REM
120 FOR S=SEG TO 1 STEP-1
130 PRINT S; "   ";
135 PAUSE 60
140 NEXT S
150 PRINT
160 PRINT
170 PRINT "FOGO... "
```

O comando PAUSE funciona até que alguma tecla seja pressionada. Se isto acontecer, e se o comando PAUSE estiver em execução (tempo de espera), o computador executa imediatamente a linha seguinte ao comando PAUSE. Evidentemente, isto não acontece com a programação de alças de espera de tempo com os comandos FOR... NEXT. O computador fica "congelado", não respondendo a nenhuma tecla (com exceção da tecla BREAK), durante todo o período de espera.

O comando PAUSE pode ser usado também para apresentar várias telas de informação, dando tempo para que o usuário as leia, para, depois, limpar a tela com o comando CLS e para apresentar em seguida nova tela...

```
10 PRINT "JOGO DE ADIVINHACAO"
20 PRINT "=================="
30 PRINT
40 PRINT "NESTE JOGO, EU VOU
     SORTEAR UM"
50 PRINT "NUMERO ENTRE 1 E 100, E
     VOCE"
60 PRINT "TEM QUE TENTAR ADIVI-
     NHAR QUAL"
70 PRINT "FOI."
80 PAUSE 300
90 CLS
100 PRINT "GANHA QUEM ADIVI-
      NHAR EM MENOR"
110 PRINT "NUMERO DE VEZES"
120 PAUSE 180
130 PRINT
  •
  •
  •
```

Neste programa, as linhas de 10 a 70 mostram a informação a ser lida na tela do vídeo. Em seguida, a linha 80 provoca uma pausa de cerca de 5 segundos (o que proporciona tempo para a leitura, mesmo para as pessoas que lêem devagar!), limpa a tela e passa para a segunda tela de informação (linhas 100 e 110), dá nova pausa (menor), na linha 120, e assim por diante.

Se a pessoa lê com rapidez, basta que ela pressione qualquer tecla ao terminar a leitura. Se o comando PAUSE correspondente ainda estiver sendo executado, ele é interrompido e passa imediatamente para a linha seguinte.

EXERCÍCIOS

1. **Faça um programa para calcular a soma dos números inteiros de 1 a 100 (faça duas versões para este programa: uma usando alças com IF, e outra usando alças com FOR... NEXT).**

2. **Modifique o programa acima, de modo a poder calcular a soma dos números inteiros entre 1 e um número qualquer (que será fornecido por um comando INPUT).**

3. **Faça um programa como o acima, só que para calcular o fatorial de um número inteiro qualquer (N!)**
   Dica: N! = 2×3×4 ...× N
   Por exemplo: 5! = 2×3×4×5 = 100
   O que acontece se eu entrar um número muito alto (por exemplo, 32 000!)?

4. **Qual é a seqüência de valores que os seguintes programas mostrarão na tela, quando executados?**

   a.
```
10 LET N=1
20 PRINT N
30 LET N=N+2
40 IF N=9 THEN STOP
50 GOTO 20
```

b.
```
10 LET D=0
20 LET I=0.25
30 LET D=D+I
40 IF D 2 THEN GOTO 70
50 PRINT D
60 GOTO 30
70 STOP
```

c.
```
10 LET N=10
20 LET N=N+1
30 IF N 20 THEN GOTO 20
40 IF N 30 THEN STOP
50 PRINT N
60 GOTO 20
```

d.
```
10 LET I= -1
20 LET S=2
30 LET S=S+1
40 IF S 10 THEN STOP
50 PRINT S
60 GOTO 30
```

5. Qual é a seqüência de valores que serão gerados pelos seguintes comandos FOR?
   a. FOR E=20 TO 100 STEP 10
   b. FOR X=10 TO 20 STEP 0.5
   c. FOR Z=5 TO 1 STEP -0.25
   d. FOR N=34 TO 36
   e. FOR N=36 TO 34
   f. FOR N=36 TO 34 STEP -1

6. Descubra e explique quais os erros nos programas seguintes:

a.
```
121 FOR X=1 TO 232 STEP 3
122 PRINT X/3
123 NEXT Y
```

Resposta:

b.
```
10 FOR X$=1 TO 25
20 NEXT X$
```

Resposta:

c.
```
300 FOR ZZ=1 45
310 NEXT ZZ
```

Resposta:

d.
```
10 FOR I= -12 TO 12 STEP 2
20 PRINT I;
30 NEXT
```

Resposta:

e.
```
10 NEXT J
20 FOR J=2 TO 5
```

Resposta:

7. Quantos segundos esperará cada comando PAUSE?
   a. PAUSE 30
   b. PAUSE 600
   c. PAUSE 10/60
   d. PAUSE 1

8. Faça um programa para aceitar um nome qualquer pelo teclado e então fazê-lo piscar por 10 vezes seguidas, no meio da primeira linha da tela. Use comandos IF.

9. Faça um programa para calcular a soma de 10 números quaisquer, digitados no teclado. Use o comando FOR.

10. Faça um programa para imprimir uma tabela das raízes quadradas dos números de 10 a 1000, de 10 em 10. Use o comando FOR.

RESPOSTAS AOS EXERCÍCIOS ANTERIORES

Página 46

7. Faça um programa para testar se o resultado da divisão de um número por outro tem resto maior do que zero.

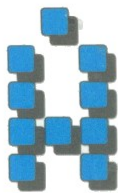
Resposta:

A função INT obtém o maior inteiro presente em um número qualquer. Quando o resto de uma divisão é zero, então o resultado da divisão é igual ao seu inteiro.

```
10 INPUT N1
20 INPUT N2
30 LET D=N1/N2
40 IF INT (D)=D THEN GOTO 60
50 PRINT "DEIXOU RESTO"
55 GOTO 10
60 PRINT "NAO DEIXOU RESTO"
70 GOTO 10
```

10. Faça um programa que imprima na tela os números de 10 a 20.

Resposta:

```
10 LET N=10
20 PRINT N
30 LET N=N+1
40 IF N < 20 THEN GOTO 20
50 STOP
```

lças dentro de alças

Muitas vezes é necessário repetir dentro de um programa alças menores dentro de alças maiores. Embora pareça complicada, esta operação é fácil de ser feita em BASIC, desde que tomemos alguns cuidados.

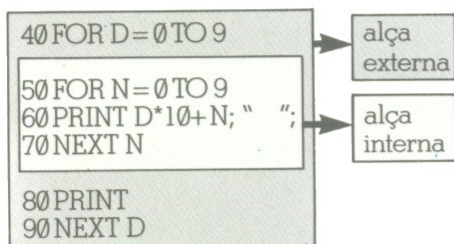
Para servir de exemplo, vamos fazer o seguinte programa:

Queremos colocar na tela a lista dos números da cartela da Loto, de modo que cada dezena comece em uma linha nova. Dessa forma, temos:

e assim por diante.

Uma maneira bem fácil de programar isto é a seguinte:

```
10 REM --- TABELA PARA APOSTAR
    NA LOTO
20 REM -----------------------------
30 REM
40 FOR D=Ø TO 9
50 FOR N=Ø TO 9
60 PRINT D*1Ø+N;"  ";
70 NEXT N
80 PRINT
90 NEXT D
100 STOP
```

Podemos ver facilmente que este programa tem duas alças, uma dentro da outra:

O primeiro FOR encontrado (linha 40) determina a repetição, por 10 vezes, de tudo que se encontra entre a linha seguinte (linha 50) e o NEXT correspondente (linha 90), ou seja, as linhas 50, 60, 70 e 80.

Entretanto, dentro dessa alça, encontramos um segundo FOR, que determina a repetição, por 10 vezes, de tudo que se encontra entre a linha seguinte (linha 60) e o NEXT correspondente (que está na linha 70).

Portanto, a linha 60:

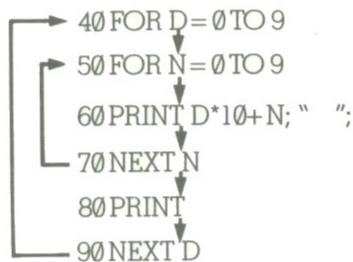
```
6Ø PRINT D*1Ø+N; "  ";
```

que é a única que se encontra dentro das duas alças, simultaneamente, será executada:

10×10= 100 vezes

Como existe um ponto e vírgula no final dessa linha, os números impressos sob controle da alça interna (linhas 50 e 70) serão colocados sucessivamente na mesma linha, um após o outro, com um espaço em branco de separação entre cada dois algarismos.

A linha 80 (que só é executada depois que a alça se completa) foi colocada para provocar uma mudança de linha no final, ou seja, para começar uma nova dezena na linha imediatamente abaixo.

Para compreendermos como duas alças podem ficar "aninhadas" isto é, uma dentro da outra, costuma-se usar o recurso das flechinhas de retorno:

Ao ser executado o programa pela primeira vez, acontece o seguinte:

1. ao encontrar a linha 40, o programa inicializa a variável contadora D com o valor inicial 0, e especifica 9 como o valor final (máximo) e 1 como o incremento (STEP);

2. ao encontrar a linha 50, o novo comando FOR inicializa a variável contadora N como o valor inicial de 0 e especifica o valor final como 9, e o incremento como 1;

3. a linha 60 é impressa pela primeira vez. Como D é zero e N também é zero, ainda teremos como resultado da expressão D*10+N o valor zero, que é impresso;

4. agora o programa encontra a linha 70 (NEXT N), que provoca o incremento da variável N de mais 1, e em seguida este valor é comparado com o valor final do índice, que é 9. Como ainda é menor do que este, a linha 60 (dentro da alça interna) é repetida mais uma vez.

   Mas agora o valor armazenado na variável N passou a ser 1, enquanto que o valor armazenado em D ainda é 0. Portanto, calcula-se:

   D*10+N é igual a 0*10+1,

   ou seja, igual a 1, que é impresso na mesma linha que o valor 0 impresso anteriormente, por causa do ponto e vírgula;

5. novamente o programa encontra a linha NEXT N, incrementa (N passa a ser 2, testa, etc.). Depois de repetir isto até que 9 seja atingido, teremos uma linha impressa assim:

6. portanto, ao terminar a alça que vai de 50 a 70, o programa executa a linha seguinte, que é 80 (PRINT), mudando de linha na tela, e cai na linha 90, que é um NEXT D, ou seja, que incrementa o valor armazenado na contadora D, testa se é menor que 9, etc. Como não é igual a 9 ainda, o programa repete a linha logo abaixo do FOR D, ou seja, a linha 50. Como esta é uma linha FOR, também repete-se tudo de novo, como acima, só que o valor de D é agora igual a 1. Portanto a linha a ser impressa passa a ser:

e assim por diante.

Vamos ver como ficariam, em uma tabela, os valores de N e D em cada repetição das duas alças:

LINHA	D	N	D*10+N
40	0	0	0
50	0	0	0
50	0	1	1
50	0	2	2
50	0	3	3
50	0	4	4
50	0	5	5
50	0	6	6
50	0	7	7
50	0	8	8
50	0	9	9
40	1	0	10
50	1	1	11
50	1	2	12
50	1	3	13
.	.	.	.
.	.	.	.
.	.	.	.

etc.

Pela tabela, você pode observar que os dois índices (D e N) das alças variam regularmente (são incrementados). Só que o índice da alça mais interna varia *mais depressa*, e o índice da alça mais externa, *mais devagar*.

Podemos colocar várias alças, umas dentro das outras.

A única regra que devemos seguir à risca é a seguinte: não pode haver "cruzamentos" entre os FOR e respectivos NEXT das alças emparelhadas. Por exemplo, o seguinte programa provocará uma mensagem de erro:

```
100 FOR I=1 TO 3
110 FOR J=1 TO 2
120 PRINT I, J
130 NEXT I
140 NEXT J
```

Se colocarmos as flechinhas que indicam os pares FOR... NEXT, veremos que há um cruzamento entre elas, o que indica o erro sintático:

```
┌─► 100 FOR I=1 TO 3
│┌► 110 FOR J=1 TO 2
││  120 PRINT I,J
└┼─ 130 NEXT I
 └─ 140 NEXT J
```

Outro arranjo não permitido é usar a mesma variável-índice para duas ou mais alças que estejam aninhadas. Embora isto, muitas vezes, não seja indicado pelo computador como um erro sintático, naturalmente pode provocar erros lógicos muito difíceis de serem detectados.

```
100 FOR I=1 TO 3
110 FOR I=1 TO 4
120 NEXT I
130 NEXT I
```

Da mesma forma, devemos evitar a modificação da variável-índice da alça dentro da mesma (por exemplo, através do comando LET ou do comando INPUT), de modo a evitar erros lógicos. Entretanto, isto é permitido pelo interpretador BASIC. Por exemplo:

```
10 FOR I=1 TO 10
20 LET I=I+1
30 PRINT I
40 NEXT I
```

Coloque este programa no computador e verifique qual é o resultado.

m programa completo

Agora você já conhece um número suficiente de comandos em BASIC para tentar seus primeiros programas completos. Para exemplificar os passos de desenvolvimento de um programa de maior porte, vamos programar juntos a seguinte tarefa:

Faça um programa para calcular a média simples das matérias para cada um dos alunos de uma classe. Ao final, o programa deve mostrar a média geral da classe. Além disso, o programa deve informar para cada aluno as suas notas, a sua média, e se foi aprovado (nota igual ou maior que 7, se ficou para recuperação (nota maior que 5 e menor que 7) ou se foi reprovado (nota menor que 5). Tanto o número de alunos quanto o número de matérias são variáveis (isto é, são diferentes para cada série, e devem ser informados ao programa).

Solução:

Vamos dividir o nosso trabalho de programação nas seguintes etapas:

a. especificação dos dados de entrada
b. especificação dos dados de saída
c. especificação das constantes internas
d. especificação das fórmulas de cálculo
e. esquema geral do programa (etapas)
f. codificação em linguagem BASIC
g. teste com alguns dados conhecidos
h. documentação

a. Especificação dos dados de entrada

Parâmetros de operação do programa:

CL$ - nome da classe (por exemplo, 5.ª série A)
NA - número de alunos na classe
NM - número de matérias que a classe tem

Dados individuais (para cada aluno)

NO$ - nome
NU - número da classe
NO - nota em uma determinada matéria

b. Especificação dos dados de saída

MG - média global da classe
MA - média geral de cada aluno

c. Especificação das constantes internas

AP - nota de aprovação (>=7)
RR - nota de recuperação (>=5 e <7)
RP - nota de reprovação (<=5)

d. Especificação das fórmulas de cálculo

S - soma das notas de cada aluno

MA = S/ND - média geral do aluno

e. Esquema geral do programa

O programa terá as seguintes partes:

1. identificação do programa
2. instruções breves de utilização
3. entrada de dados paramétricos (nome da classe, número de alunos e de matérias)
4. alça de entrada dos alunos: entrada do número de classe e nome

5. alça de entrada das notas de cada aluno
6. cálculo da média do aluno
7. acumulação da soma das médias da classe
8. impressão dos resultados de um aluno
9. cálculo e impressão da média global
10. fim do programa

f. Codificação em linguagem BASIC

```
10 REM ----------------
20 REM - MEDIAS 1.00 JAN 85
30 REM - CALCULO DE MEDIAS DE UMA
40 REM - CLASSE P/ SINCLAIR
50 REM - (C) 1985 R. SABBATINI
60 REM ----------------
70 REM
80 REM --- 1. IDENTIFICACAO ----
90 REM
100 PRINT TAB 12; "MEDIA 1.00 "
110 PRINT "CALCULO DAS MEDIDAS DA CLASSE"
120 PRINT
130 PRINT
140 REM
150 REM --- 2. OBJETIVO ----
160 REM
170 PRINT "ESTE PROGRAMA TEM POR OBJETIVO"
180 PRINT "CALCULAR AS MEDIAS GERAIS
    DOS ALUNOS "
190 PRINT "DE UMA CLASSE, A PARTIR DAS"
200 PRINT "SUAS NOTAS NAS DISCIPLINAS."
210 PRINT
220 PRINT "FORNECA OS SEGUINTES DADOS"
230 PRINT "PARA A CLASSE:"
240 PRINT
250 REM
260 REM --- 3. DADOS PARAMETRICOS ----
270 REM
280 PRINT "NOME DA CLASSE?"
290 INPUT CL$
300 PRINT "NUMERO DE ALUNOS NA CLASSE?"
310 INPUT NA
320 NA = INT (NA)
330 IF NA < 1 THEN GOTO 310
340 PRINT "NUMERO DE DISCIPLINAS?"
350 INPUT NM
360 NM = INT (NM)
370 IF NM < 1 THEN GOTO 350
```

```
375 LET MG = 0
380 REM
390 REM --- 4. ENTRADA DOS ALUNOS
400 REM
405 FOR ALUNO = 1 TO NA
410 CLS
420 PRINT "CLASSE : "; CL$ ;" - ";
430 PRINT "ALUNO NO."; ALUNO
440 PRINT " = = = = = = = = = = = = = = = = = = "
450 PRINT
460 PRINT "DIGITE OS DADOS DESTE ALUNO:"
470 PRINT
480 PRINT "NOME?"
490 INPUT NO$
500 PRINT "NUMERO DA CLASSE?"
510 INPUT NU
520 PRINT
530 REM
540 REM --- 5. ENTRADA DAS NOTAS ---
550 REM
560 LET S = 0
570 FOR NOTA = 1 TO NM
580 PRINT "NOTA DA MATERIA" ; NOTA
590 INPUT NO
600 IF  NO < 0  THEN 590
610 IF  NO > 10 THEN 590
620 LET S = S + NO
630 NEXT NOTA
635 LET MA = S/NM
640 PRINT
650 PRINT "MEDIA GERAL = "; MA
660 LET MG = MG + MA
670 NEXT ALUNO
680 REM
690 REM ---- 6. CALCULO MEDIA GLOBAL ---
700 REM
710 CLS
720 PRINT "MEDIA DA CLASSE" ; MG/NA
730 STOP
```

TESTE

9.1. Quando é necessário usar variáveis dentro de um programa para armazenar o resultado de expressões?

Resposta: ______________________________

____

9.2. Indique o que está errado nas linhas abaixo:

a. 25 LET NOME$ = 25 ______________

b. 30 LET VARI$ = "12AB ______________

c. 40 LET C = "COMPUTADOR" ______________

d. 50 LET C$ = 25 + 4*22 ______________

e. 60 LET LOG(N) = 423 ______________

f. 70 PRINT X = 225 ______________

g. 80 LET "COMPUTADOR" = C$ ______________

9.3. Identifique, na coluna à direita, qual é o tipo de constante ou variável em BASIC, marcando a letra correspondente na coluna à esquerda:

a. variável alfanumérica	( ) 123.14
b. variável numérica	( ) "ATOMO"
c. constante alfanumérica	( ) "123"
d. constante alfanumérica	( ) v
e. comando BASIC	( ) LET
	( ) ABC$
	( ) "LET"
	( ) "PRINT 123"
	( ) RUN 100

9.4. Como faz o computador para armazenar na memória um nome composto de vários caracteres?

Resposta: ______________________________

9.5. Defina:

a. caractere: ______________

b. cadeia de caracteres: ______________

c. variável alfanumérica: ______________

d. variável numérica: ______________

e. constante alfanumérica: ______________

f. constante alfanumérica: ______________

9.6. Qual é o comprimento máximo de uma cadeia de caracteres permitido pelo BASIC?

Resposta: ______________________________

9.7. Descreva as diferenças básicas de funcionamento entre os comandos LET e INPUT. O que têm eles em comum?

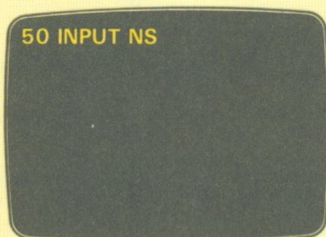
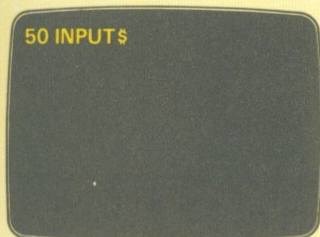
Resposta: ____

____

____

9.8. Mostre como aparecerá a linha de baixo da tela, quando o programa encontrar os seguintes comandos (dica: cursor K ou L, com ou sem aspas):

50 INPUT$

50 INPUT N$

9.9. Como se deve proceder para interromper a execução do programa, caso ele esteja parado nas linhas acima?

Resposta: ____

____

9.10. Cite três usos para o comando REM em um programa.

Resposta: 1. ____

2. ____

3. ____

9.11. O que é um comando executável? Dê três exemplos.

Resposta: 1. ____

2. ____

3. ____

9.12. Identifique na coluna à esquerda cada comando em BASIC, com a função que ele realiza na coluna à direita:

a. RUN	( ) armazena algo em uma variável
b. NEW	( ) linha de comentário
c. STOP	( ) inicia a repetição de uma alça
d. PRINT	( ) executa um programa
e. LET	( ) continua um programa interrompido
f. REM	( ) mostra um resultado na tela
g. INPUT	( ) testa uma condição
h. PAUSE	( ) limpa a tela
i. GOTO	( ) continua a repetição de uma alça
j. CLS	( ) interrompe a execução do programa
l. FOR	( ) apaga um programa da memória
m. NEXT	( ) entra um valor pelo teclado
n. LIST	( ) desvia o programa para outra linha
o. CONT	( ) lista um programa na tela
p. EDIT	( ) corrige uma linha de programa
q. IF	( ) provoca uma pausa no programa

9.13. Defina:

a. variável somadora: ______________________

b. variável contadora: ______________________

c. alça de repetição: ______________________

d. cláusula lógica: ______________________

e. operador relacional: ______________________

f. alça sem fim: ______________________

g. aninhamento de alças: ______________________

9.14. Identifique os elementos básicos de uma alça:

FOR I = 1 TO 12 STEP 2

9.15. O que o programa abaixo vai mostrar na tela ao ser executado?

```
10 FOR I=1 TO 100 STEP 10
20 FOR J=I TO I+2
30 PRINT J;
40 NEXT J
50 NEXT I
```

Resposta:

9.16. O que acontecerá ser for adicionada a seguinte linha ao programa acima?

45 PRINT

9.17. Faça uma programa para gerar na tela todas as combinações possíveis de preenchimento do cartão da Loteria Esportiva. Por exemplo:

1 - COLUNA 1
2 - COLUNA 1
    COLUNA 2
3 - COLUNA 1
    COLUNA 3

Dica: numere as colunas 1, 2 (do meio) e 3.

9.18. Cite duas regras importantes que não podem ser violadas no aninhamento de alças.

Resposta: ______________________

____

Cite duas vantagens do comando FOR para construir alças, em relação ao comando IF.

Resposta: ______________________

____

9.19. Modifique o programa para o cálculo das médias de uma classe (páginas 54 e 55), de modo a permitir colocar as notas-limites para aprovação, recuperação e reprovação, por meio de comandos INPUT (variáveis).

9.20. Faça um programa para apresentar na tela três exercícios de tabuada de multiplicar e testar se o resultado apresentado pelo aluno foi correto. O programa deve dizer, ao final, quantos exercícios foram resolvidos corretamente pelo aluno.

C A P Í T U L O 7

ORTE E AZAR NO COMPUTADOR

Para muitas pessoas, o jogo é uma das coisas mais interessantes e divertidas que se pode fazer com um computador. Se o jogo incluir efeitos de acaso, sorte ou azar, melhor ainda!

Este capítulo será quase todo dedicado aos jogos e aos modos de programá-los. Vamos começar com um joguinho bem simples, de adivinhação, depois estudar como fazer um programa útil ao ensino e, finalmente, vamos programar alguns joguinhos que utilizam gráficos na tela.

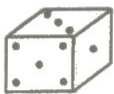
Para introduzir o acaso (sorteio de um dado, por exemplo) em um jogo, é necessário conhecer uma função própria da linguagem BASIC que permite sortear números.

Esta função é chamada de RND, abreviatura da palavra inglesa RANDOM, ou seja, *aleatório* (ao acaso).

A função RND, portanto, serve para sortear *números aleatórios*.

Números aleatórios são números escolhidos ao acaso pelo computador, a partir de certo conjunto maior de números.

Vamos imaginar, por exemplo, como é o sorteio de um número em um dado.

Ao lançarmos o dado, a face que ficar para cima será o número sorteado.

Naturalmente só temos seis alternativas possíveis no lançamento do dado. A este conjunto de números chamamos de *espaço amostral*. O número aleatório só poderá ser um destes seis.

Outro exemplo seria uma moeda, no jogo de cara ou coroa. Só temos aqui duas alternativas. Vamos excluir a possibilidade de a moeda ficar de pé sobre a borda! O espaço amostral da moeda é igual a 2. O espaço amostral do dado é igual a 6.

A probabilidade de que um determinado número aleatório saia em um sorteio é definida como o inverso do número das alternativas possíveis.

Assim, por exemplo, a probabilidade de sair no dado um determinado número é de:

$$\frac{1}{6} \quad \frac{\text{um evento em}}{\text{seis alternativas possíveis}}$$

Expressando em percentagens, haveria a probalidade de sair um número qualquer como sendo:

$$\frac{100 \times 1}{6} = 16,666\dot{6}\%$$

Em outras palavras, se jogarmos um dado 100 vezes, cada uma das faces será sorteada cerca de 16 vezes aproximadamente. Para calcular, portanto, quantas vezes se espera um determinado resultado, multiplicamos a probabilidade pelo número de vezes em que ser realizará o sorteio.

Quantas vezes, por exemplo, se espera que saia coroa, se lançarmos uma moeda 200 vezes?

$1/2 \times 200 = 0,5 \times 200 = 100$ vezes

1/2 é a probabilidade de que coroa seja sorteada um um lançamento.

Note que a probabilidade de sair cara é a mesma de sair coroa, ou seja, 50%. Tanto neste caso, como no caso de um dado, dizemos então que a *distribuição probabilística* e UNIFORME, ou seja, todos os números possíveis têm a mesma probabilidade de ser sorteados.

A função RND

Como podemos utilizar esta função do BASIC para simular o sorteio de números aleatórios uniformes?

Você se lembra de que uma função em BASIC efetua sempre um cálculo ou devolve um resultado, a partir de um dado ou argumento que dermos a ela dentro do programa.

Para calcularmos, por exemplo, a raiz quadrada do número 25, fazemos assim:

```
LET R = SQR (25)
```

sendo 25 o dado ou argumento. A função SQR, quando chamada, retorna o resultado (no caso, 5), que é armazenado na variável indicada por LET.

Podemos também colocar a função dentro de uma expressão:

```
LET RAIZ = 100 * SQR (25)/3
```

ou ainda, dentro de um comando PRINT:

```
PRINT SQR (25)
```

A função RND é um pouco diferente. Ela não precisa de um argumento. Bastaria colocar, por exemplo,

```
PRINT RND
```

para sortear um número aleatório ou usar RND dentro de uma expressão LET.

Em segundo lugar, a função RND não faz nenhum cálculo matemático, como no caso da raiz quadrada (SQR) e outras funções em BASIC deste tipo.

Ela apenas retorna um número aleatório de distribuição uniforme, compreendido entre zero (0) e um (1).

Cada vez que a função RND é usada, um nmero aleatório inteiramente diferente é sorteado.

Por isso, a função RND è chamada de *geradora de números aleatórios*.

O conceito de número aleatório, no caso da função RND é parecido com o do sorteio de dados e moedas, só que o número sorteado não é um número inteiro (1, 2, 3 etc.), mas um número fracionário, entre 0 e 1.

Se colocarmos, por exemplo, no computador o seguinte programa:

```
10 FOR I=1 TO 10
```

```
20 PRINT RND
30 NEXT I
```

sortearemos em seguida 10 números aleatórios, que serão impressos na tela assim:

```
0.03892129
0.12455412
0.67129093
0.00112232
0.48983312
0.87811121
0.11201232
0.00989823
0.01909123
0.53219883
```

Se você executar novamente o programa, um novo conjunto de 10 números totalmente diferente do primeiro será mostrado na tela. Um número deste tipo nunca se repete.

Na realidade, o nosso "sorteador" não é tão perfeito assim. O número 1 "redondo" (1.000000000) nunca é sorteado. Os limites do sorteio caem então entre 0.000000000 (zero perfeito) e 0.999999999.

O gerador de números aleatórios do BASIC é do tipo *universal*, ou seja, ele pode ser usado para simular qualquer tipo de sorteio (dados, moedas, baralhos etc.). Vamos ver como isto é feito.

Comecemos com o caso mais simples: a moeda.

Cada evento probabilístico (em outras palavras, o que pode resultar do lançamento honesto de uma moeda) tem a probabilidade de 0,5.

Portanto, para simular o lançamento de uma moeda com a função RND, podemos fazer simplesmente isto:

1. sorteamos um número fracionário de 0 a 1 com a função RND;

2. testamos se este número é menor do que 0,5. Se for, consideramos que saiu cara, por exemplo. Se for maior do que 0,5, consideramos que saiu coroa (podia ser o contrário, pois esta é uma convenção que depende apenas do programador).

Vejamos um programa que realiza isto:

```
10 LET R=RND
20 IF R<0.5 THEN PRINT
   "CARA"
30 IF R>=0.5 THEN PRINT
   "COROA"
40 GOTO 10
```

Vamos interpretar o que faz este programa:

- linha 10 = sorteie um número fracionário entre 0 e 1 e armazene-o na variável R;

- linha 20 = se R for menor do que 0.5, imprima a palavra CARA;

- linha 30 = se R for maior ou igual a 0.5, imprima a palavra COROA;

- linha 40 = volte para a linha 10 (novo sorteio).

Se executarmos este programa, poderemos ter o seguinte resultado:

```
CARA
COROA
CARA
CARA
CARA
CARA
COROA
CARA
COROA
COROA
```

e assim por diante, até encher totalmente a tela. Observe que, neste sorteio, saíram apenas três coroas e cinco caras. Mas não havíamos dito que o número de caras e coroas seria igual?

Bem, isto só vai acontecer se sortearmos um número muito grande de vezes! Quanto mais sortearmos, mais semelhante será a ocorrência dos dois eventos. Esta é a chamada *lei dos grandes números*.

Como a função RND tornou possível um sorteio "honesto" da moeda?

Você se lembra de que nós dissemos que a distribuição aleatória do gerador RND é uniforme, não é?

Isto significa simplesmente que cada um dos números a serem sorteados tem a mesma probabilidade. Em conseqüência, se dividirmos este conjunto de números em duas partes (menor e maior do que 0,5 respectivamente), a probabilidade de sair um número que caia numa destas duas metades continua sendo igual, certo?

Como dividimos o espaço amostral em apenas duas possibilidades, será de 0,5 a probabilidade de cada uma delas!

Podemos estender este raciocínio para outras divisões: em 3, 4, 5, 10 possibilidades etc.

Mas há um inconveniente: teremos que aumentar o número de IF no programa, para poder testar todas as possibilidades.

Existe, felizmente, um meio mais fácil de fazer isto. Usaremos um artifício matemático muito simples e fácil de entender, que serve para qualquer ocasião.

Vamos agora tomar como exemplo o dado.

O dado tem um espaço amostral de 6: existem apenas seis possibilidades de sorteio.

Se multiplicarmos o resultado da função RND por 6, teremos o seguinte:

menor número que pode ser sorteado por RND $=0$
multiplicado por 6 $=6\times0=0$

maior número que pode ser sorteado por RND $=0.999999999$
multiplicado por 6 $=5.999999994$

Teremos, portanto, um número fracionário, que pode ser desde 0 até 5.999999994

Como queremos um número inteiro distribuído entre 1 e 6, basta usarmos a seguinte expressão em BASIC:

```
INT (RND*6) + 1
```

Vamos analisar o que esta expressão faz com o número sorteado:

- primeiro, ela multiplica o número aleatório por 6, dando, portanto, um número aleatório entre 0 e 5.999999994;

- em seguida, calcula o inteiro deste número (ou seja, o dígito à esquerda do ponto) e obterá um número de 0 a 5;

- finalmente, soma 1 a este número. Portanto, teremos um número de 1 a 6.

Conseguimos programar um sorteador de dados!

Vamos tomar um exemplo com alguns números aleatórios, para entender como funcionam estas transformações:

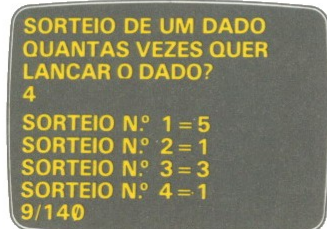
RND	RND * 6	INT (RND * 6)	INT (RND*6) + 1
0.0000000	0.0000000	0	1
0.0025342	0.0152052	0	1
0.3451235	2.070741	2	3
0.9999999	5.9999994	5	6

Para ver um maior número de exemplos, coloque no computador o programa abaixo:

```
10 LET R = RND
20 PRINT R; R*6; INT (R*6); INT
   (RND*6) + 1
30 GOTO 10
```

Agora podemos programar o nosso primeiro joguinho, para sortear um dado quantas vezes quisermos:

```
10 REM --- JOGO DE SORTEIO DE
   UM DADO
20 REM --- PARA MICROS TIPO ZX
30 REM -----------------------------------
40 REM
50 PRINT "SORTEIO DE UM DADO"
60 PRINT
70 PRINT "QUANTAS VEZES QUER
   LANCAR O DADO?";
80 INPUT NV
90 PRINT NV
100 PRINT
110 FOR I = 1 TO NV
120 PRINT "SORTEIO NO."; I;" = ";
    INT (RND*6) + 1
130 NEXT I
140 STOP
```

Um exemplo de jogada seria o seguinte:

Teoricamente, um bom gerador de números aleatórios nunca deveria repetir a mesma seqüência de números sorteados. Entretanto, isto é impossível de se conseguir na prática, devido à maneira como eles são programados na memória ROM do computador.

Explicando de maneira simplificada, um gerador interno de números aleatórios funciona da seguinte maneira:

Um número aleatório, ao ser gerado pela função RND, fica armazenado internamente em uma memória de trabalho. Uma segunda vez que a função RND for chamada, ela transforma este número armazenado por um processo matemático especial, para gerar um outro número aleatório. Este processo se repete tantas vezes quantas a função RND for chamada.

Por isso, dizemos que o gerador aleatório precisa de uma "semente" para funcionar. O número gerado em cada chamada serve de semente para o próximo. A seqüência então é pseudo-aleatória, e depois de alguns milhões de sorteios pode se repetir novamente.

O mais grave, entretanto, é que, quando ligamos o computador pela primeira, vez, o gerador aleatório sempre usa uma "semente" fixa, que é 0.0022735596.

Isto significa que o joguinho que você programar poderá estar "viciado", desde que se saiba qual é a seqüência inicial.

Para evitar isto, o BASIC tem um comando chamado RAND, que pode ser usado logo no começo do programa, para gerar uma "semente" ao acaso, ou seja, a primeira semente da série de sorteios.

Para todos so efeitos, precisamos dar um comando RAND ao computador apenas uma vez antes de cada série de sorteios.

Por exemplo:

```
5 RAND
10 PRINT INT (RND*2) + 1;
20 GOTO 10
```

vai sortear números 1 e 2 ao acaso. A linha 5, com o comando RAND, será uma sementinha aleatória inicial, de modo que ninguém conseguirá advinhar a seqüência de sorteios.

Note bem: não confunda o *comando* RAND com a *função* RND.

EXERCÍCIOS

1. Faça um programa para simular várias vezes o sorteio de uma moeda (o usuário informa quantas vezes ele quer).

2. Modifique o programa acima para contar quantas vezes saiu cara e quantas vezes saiu coroa, informando o resultado ao final:

```
SORTEIO DE UMA
MOEDA 45 VEZES

SAIRAM 19 CARAS
(42.22222%)

SAIRAM 26 COROAS
(57.77777%)
```

3. Indique qual o número aleatório mínimo e máximo que podem sair, em função das expressões abaixo:

	Mínimo	Máximo
a. INT (RND)	______	______
b. INT (RND*8) + 2	______	______
c. INT (RND*2)-1	______	______
d. RND/2	______	______
e. RND*RND	______	______

4. Teste no computador qual será o resultado dos seguintes programas:

a.
```
10 PRINT RND (0)
20 GOTO 10
```

b.
```
10 PRINT RND (1)
20 GOTO 10
```

5. Faça um programa para sortear as cartas de um baralho, mostrando o seu valor ou figura na tela.

Dica: as cartas de um baralho são: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K e A, em quatro naipes (espadas, copas, paus e ouros).

Faça dois sorteios no programa: um para sortear a figura, outro para sortear o naipe. Depois apresente o resultado assim:

A CARTA SORTEADA
FOI 5 DE COPAS.

A CARTA SORTEADA
FOI VALETE DE PAUS.

e assim por diante.

6. Faça um programa para imprimir 20 sorteios de um número aleatório R, calculando é imprimindo em cada sorteio as seguintes transformações do programa:

a. R*10
b. 1 + INT (R*10)
c. 10 + INT (R*10)

7. O que fazem os programas abaixo?

a.
```
10 FOR I=1 TO INT (RND*200)
20 NEXT I
30 CLS
40 PRINT I
50 LET I=I+1
60 GOTO 10
```

b.
```
10 LET T=1+INT (RND*30 )
20 PRINT TAB T;"*";
30 CLS
40 GOTO 10
```

8. Escreva as expressões com a função RAND, necessárias para sortear números aleatórios que simulem os seguintes jogos:

a. roleta (números 0 a 35)________

b. palitinho (0 a 5 palitos)________

c. loto (números 0 a 99)________

9. Faça um programa que sorteie 10 vezes um número aleatório inteiro, compreendido entre um mínimo de 50 e um máximo de 100 (inclusive).

10. Qual é o número esperado de vezes que:

a. sairá o número 6 em um dado lançado 300 vezes? ________

b. sairá o ás em 20 cartas retiradas de 20 baralhos? ________

Jogos interativos

Um jogo que só pode sortear coisas não é muito interessante, pois logo a gente se cansa. Além disso, é mais divertido usar uma moeda, um dado ou um baralho de verdade!

A grande vantagem que o computador oferece na programação de jogos é sua interação com a pessoa. Podemos fazer com que o computador jogue com uma pessoa para ver quem ganha.

Programar um jogo deste tipo é fácil. Basta usar os comandos:

RND - para sortear alguma coisa
INPUT - para pegar respostas (ou jogadas) do usuário.
IF - para testar se é igual ou diferente do sorteado

Vamos começar com um jogo bem simples: o tradicional cara ou coroa.

OBJETIVO DO JOGO
"CARA OU COROA" Nº 1:

O computador pergunta à pessoa o que irá aparecer: cara ou coroa. Em seguida, sorteia a moeda e mostra se a pessoa acertou. Se ela errou, quem ganha é o computador. Se ela acertou, o computador perde.

```
10  REM --- JOGO DE CARA OU
    COROA Nº 1
20  REM ---------------------------
30  REM
40  RAND
50  PRINT "CARA OU COROA"
60  PRINT "============="
70  PRINT
80  PRINT "ESCOLHA O QUE VO-
    CE QUER:"
90  PRINT
100 PRINT "1 = CARA 2 = COROA"
110 PRINT
120 PRINT "DIGITE UM NUMERO 1
    OU 2"
130 INPUT JOGO
140 IF JOGO < 1 THEN GOTO 120
150 IF JOGO > 2 THEN GOTO 120
160 REM
170 REM --- SORTEIO DA MOEDA
180 REM
190 LET FACE = 1 + INT (RND*2)
200 PRINT
210 IF FACE = 1 THEN PRINT "SAIU
    CARA"
220 IF FACE = 2 THEN PRINT "SAIU
    COROA"
230 REM
240 REM --- VERIFICA QUEM GA-
    NHOU
250 REM
260 PRINT
270 IF JOGO = FACE THEN 300
280 PRINT "EU GANHEI"
290 GOTO 310
300 PRINT "VOCE GANHOU"
310 STOP
```

Para jogar outra partida, digite RUN novamente ou então modifique o programa, substituindo a linha 310 por

```
310 GOTO 70
```

Vamos fazer agora um jogo um pouco mais sofisticado:

OBJETIVO DO JOGO
"CARA OU COROA" Nº 2:

O computador pergunta à pessoa o que irá aparecer: cara ou coroa. Em seguida, sorteia a moeda e mostra se

a pessoa acertou. Os sorteios são repetidos *n* vezes. Ganha quem acertar maior número de vezes.

Não precisa apagar da memória do computador o jogo anterior. Basta modificar ou inserir as linhas que são diferentes na listagem abaixo:

```
10  REM --- JOGO DE CARA OU
    COROA Nº 2
20  REM -----------------------------
30  REM
40  RAND
42  LET NCOMP=0
44  LET NJOG=0
50  PRINT "CARA OU COROA"
60  PRINT "=========="
62  PRINT "QUANTAS VEZES
    QUER JOGAR ?"
64  INPUT NV
66  FOR N=1 TO NV
70  PRINT
75  PRINT "JOGADA Nº" ; N
80  PRINT "ESCOLHA O QUE VO-
    CE QUER :"
90  PRINT
100 PRINT "1 = CARA 2 = CO-
    ROA"
110 PRINT
120 PRINT "DIGITE UM NÚMERO 1
    OU 2"
130 INPUT JOGO
140 IF JOGO < 1 THEN GOTO 120
150 IF JOGO > 2 THEN GOTO 120
160 REM
170 REM --- SORTEIO DA MOEDA
180 REM
190 LET FACE=1 + INT (RND * 2)
200 PRINT
210 IF FACE=1 THEN PRINT "SAIU
    CARA"
220 IF FACE=2 THEN PRINT "SAIU
    COROA"
230 REM
240 REM --- VERIFICA QUEM GA-
    NHOU
250 REM
260 PRINT
270 IF JOGO=FACE THEN 300
280 PRINT "EU GANHEI"
285 LET NCOMP=NCOMP+1
290 GOTO 310
300 PRINT "VOCE GANHOU"
305 LET NJOG=NJOG+1
310 NEXT N
320 REM
330 REM --- VERIFICA NUMERO FI-
    NAL DE PONTOS
340 REM
350 PRINT "EU ACERTEI";
    NCOMP;" VEZES"
360 PRINT "VOCE ACERTOU"
    ;NJOG;" VEZES"
370 PRINT
380 IF NJOG >NCOMP THEN
    PRINT "PARABENS, VOCE GA-
    NHOU"
390 IF NJOG < NCOMP THEN
    PRINT "EU SOU O CAMPEAO"
400 IF NJOG=NCOMP THEN
    PRINT "EMPATE"
410 STOP
```

As diferenças entre este programa e o número 1 são as seguintes:

- a variável NCOMP, zerada na linha 42, servirá para contar o número de vitórias do computador;

- a variável NJOG, zerada na linha 44, servirá para contar o número de vitórias do jogador;

- a variável NV indica o número de sorteios que serão realizados e que controla a alça que vai da linha 66 até a linha 310

- nas linhas 285 e 305 são incrementadas as contadoras NCOMP e NJOG, conforme o ganhador tenha sido o computador ou o jogador, respectivamente;

- as linhas 320 a 400 servem para anunciar o resultado final.

Com alguma imaginação você conseguirá aperfeiçoar este jogo até que ele se torne realmente emocionante.

EXERCÍCIOS

1. Faça um programa de cara ou coroa semelhante ao anterior, usando, porém, um dado, ao invés de uma moeda.

Dicas: Na linha 130 a pessoa deve entrar um número de 1 a 6. Modifique as linhas anteriores (para instruções) e a linha 150 (teste):

- **modifique a expressão da linha 190;**
- **modifique as linhas 210 e 220 que anuciam o resultado;**
- **modifique as linhas de título.**

2. Faça um programa que sorteie dois dados, e mostre a soma dos números sorteados. Se a soma for igual a 7, pare de sortear. Se for diferente de 7, sorteie novamente.

3. Faça um programa que sorteie as cartas de um baralho completo, perguntando antes ao jogador qual naipe ele acha que sairá sorteado.

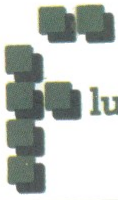
4. Explique por que o seguinte programa pode ser usado para substituir o comando RAND, em computadores que não o possuem:

```
10 PRINT "QUE HORAS SAO
   (HH.MM)"
20 INPUT H
30 FOR I=1 TO INT (H*10 )
40 LET R=RND
50 NEXT I
```

5. Faça um programa para simular o seguinte jogo ("Porquinho"): o computador sorteia um dado várias vezes. O número de pontos de cada lançamento é acumulado com os dos lançamentos anteriores. Depois de cada lançamento, o programa pergunta ao jogador se quer continuar ou desistir. Se o jogador desistir, fica com o número acumulado de pontos até aquele momento. Se continuar, entretanto, caso o computador sorteie o número 1, perde tudo o que já acumulou antes. Ganha quem acumular mais pontos.
Dica: Use um acumulador chamado SOMA. No final de cada sorteio, pergunte:

QUER CONTINUAR (SIM OU NAO)?

e verifique se a resposta foi "SIM" ou "NAO".

Fluxogramas

Nesta seção, vamos aprender uma nova maneira de planejar programas. Você já deve ter notado que, quando o programa tem uma lógica muito complexa, ou seja, tem muitos desvios condicionais e incondicionais (IF, GOTO etc.) fica muito difícil visualizar quais os caminhos que o programa deve seguir.

O planejamento prévio do programa, nestes casos, é uma atividade muito importante, porque auxilia o programador a desenvolver programas:

- mais facilmente escritos na linguagem BASIC;
- mais livres de erros, ou seja, com pouco ou nenhum erro, quando testados pela primeira vez;
- cujos erros são facilmente encontrados, corrigidos etc.;
- mais simples e compactos, ou seja, facilmente compreensíveis, eficientes e que ocupam um espaço menor de memória.

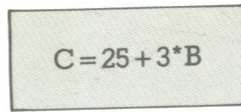
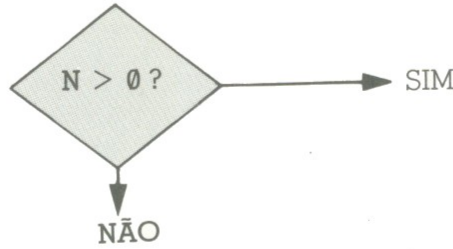
Se digitamos um programa no computador, à medida que as idéias vêm à nossa cabeça, há uma grande probabilidade de que este programa fique confuso, com erros e de difícil compreensão. Embora muitas vezes ele faça corretamente o trabalho para o qual foi destinado, nem sempre o faz da forma mais eficiente.

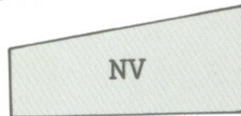
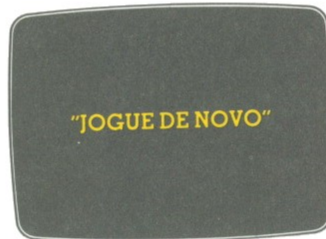
Tudo isto é especialmente mais evidente em relação aos programas com lógica mais complexa.

Um modo eficaz de planejar programas deste tipo é o denominado FLUXOGRAMA, também chamdo de DIAGRAMA DE BLOCO.

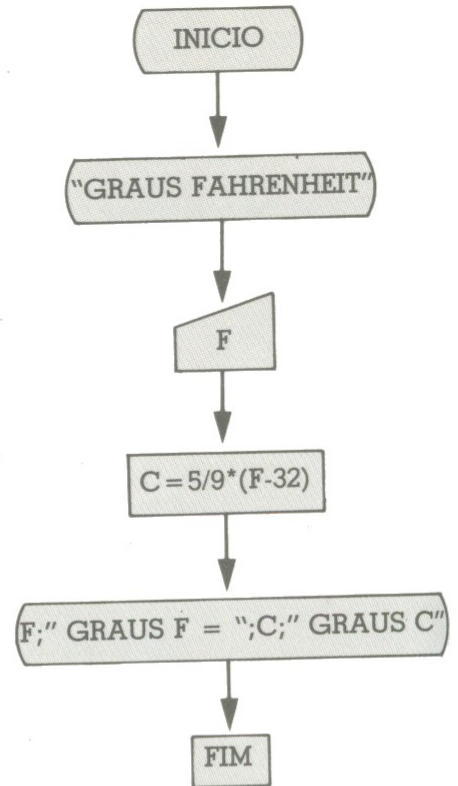
O fluxograma é uma forma de representar graficamente (por símbolos ou figuras) a estrutura e o *fluxo* de um programa.

Cada parte do programa (geralmente cada linha) é representada por um símbolo especial, que corresponde estreitamente ao tipo de comando ou instrução a ser usado. Estes símbolos são denominados *blocos*.

Por exemplo, o comando LET é representado por uma "caixinha", ou retângulo, que contém a expressão ou atribuição a ser realizada:

O comando IF é representado por um lozango que contém a condição:

O comando STOP é representado por esta figura:

O comando INPUT é representado pelo perfil de um teclado que traz o nome da variável:

E o comando PRINT é representado por uma tela de vídeo:

Existem muitos outros símbolos que posteriormente conheceremos.

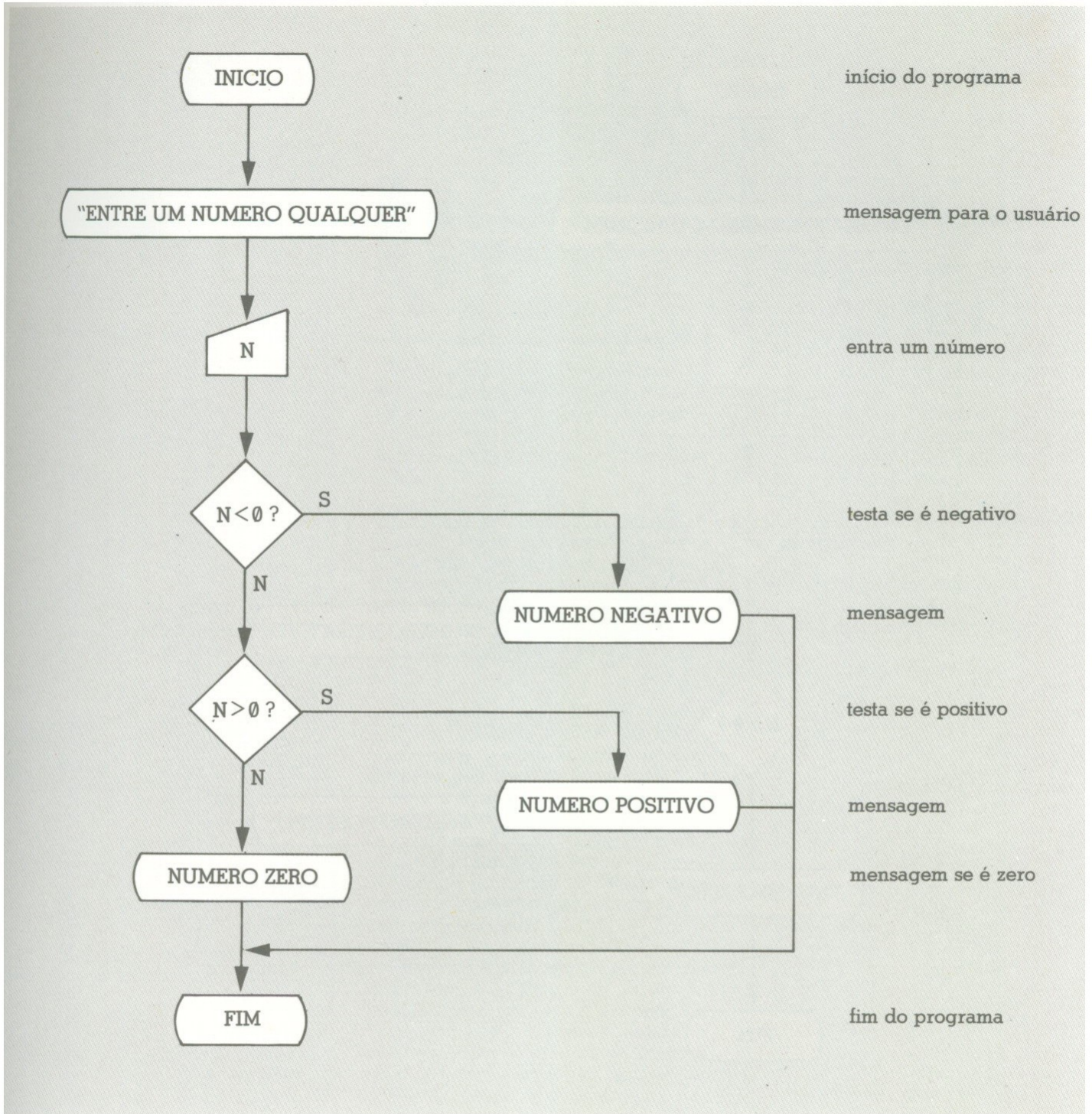
Estes símbolos são dispostos no papel em seqüência, de cima para baixo, representando os sucessivos comandos que comporão o programa. Para indicar o fluxo ou direção de um comando para outros, usam-se sinais em forma de seta.

Muitos programas têm uma *lógica linear*. (Isto quer dizer que eles não têm bifurcações ou desvios através de IFs ou GOTOs. É o caso do programa abaixo, que serve para converter graus Fahrenheit em Celsius (escalas de medida de temperatura):

Observe que todo fluxograma tem obrigatoriamente um INÍCIO e um FIM, marcados pelos símbolos correspondentes. Eles não precisam ser necessariamente representados na listagem, como abaixo:

```
10 PRINT "GRAUS FAHRENHEIT?"
20 INPUT F
30 LET C=5/9*(F-32)
40 PRINT F; "GRAUS F = "; C;"
   GRAUS C"
```

Vamos ver agora um exemplo completo de programação bem simples, para entender como é feito o fluxograma para programas com desvios.

Vamos fazer um programa para testar se um determinado número digitado pelo usuário é negativo, positivo ou zero (já estudamos este exemplo no capítulo anterior, às páginas 37 e 38).

O fluxograma ficaria assim (veja os comentários à direita):

Como vemos, é bastante fácil visualizar todas as alternativas a cada teste. Cada teste tem apenas duas saídas: verdadeira (SIM), que geralmente é colocada à direita do símbolo do IF, ou falsa (NÃO), que é colocada logo abaixo.

Para iniciar a codificação em BASIC, basta numerar os símbolos seqüencialmente, prestando atenção aos desvios. A numeração é colocada ao lado de cada símbolo:

INICIO

10 "ENTRE UM NUMERO QUALQUER"

20 N

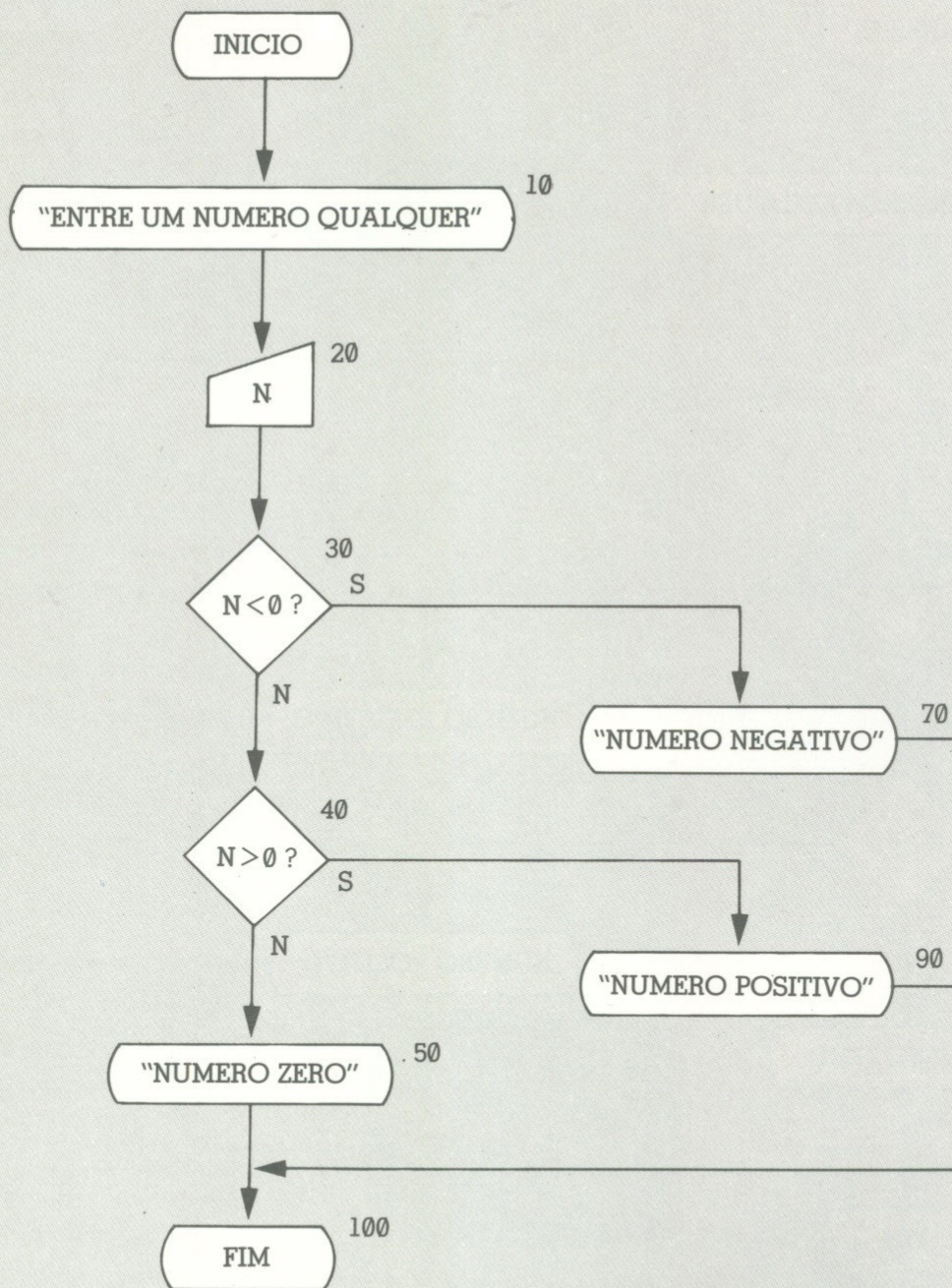
30 N<0 ? S / N

70 "NUMERO NEGATIVO"

40 N>0 ? S / N

90 "NUMERO POSITIVO"

50 "NUMERO ZERO"

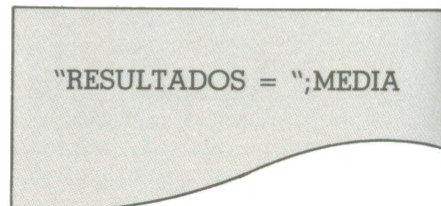
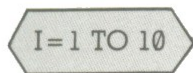
100 FIM

Agora fica muito mais fácil escrever o programa:

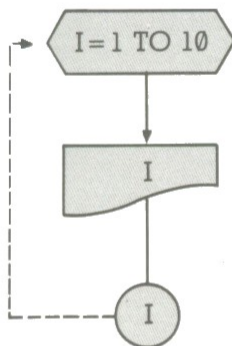
```
10 PRINT "ENTRE UM NUMERO
   QUALQUER"
20 INPUT N
30 IF N<0 THEN GOTO 70
40 IF N>0 THEN GOTO 90
50 PRINT "NUMERO ZERO"
60 GOTO 100
70 PRINT "NUMERO NEGATIVO"
80 GOTO 100
90 PRINT "NUMERO POSITIVO"
100 STOP
```

Observe que foi necessário usar apenas dois comandos IF para testar três condições. Os comandos GOTO das linhas 60 e 80 têm sua necessidade deduzida a partir do exame do fluxograma, e não precisam ser especificados nele (embora não seja proibido).

O símbolo do comando GOTO é um círculo pequeno:

Outros símbolos bastante usados para fluxogramas de programação:

Impressão em impressora (parece uma folha de papel rasgada):

O comando FOR é representado por esta figura:

O comando NEXT correspondente tem o mesmo símbolo do GOTO:

Eles são ligados por uma linha interrompida:

Se o desenho não cabe em uma única folha de papel, podemos usar o seguinte símbolo para indicar uma *continuação* ou *mudança de página*:

Com o tempo, você terá oportunidade de usar todos estes símbolos.

Uma boa idéia é adquirir uma *régua* ou *gabarito de fluxogramas*, em papelarias ou casas especializadas em materiais para desenho. Ela já tem os elementos vazados para os símbolos de programação

De agora em diante, procure sempre desenhar um fluxograma para cada programa, antes de tentar escrevê-lo em BASIC.

Isto vai aumentar muito a qualidade e correção de seus programas.

Um jogo de adivinhação

Você agora tem a possibilidade de programar uma variedade enorme de jogos de sorte ou azar. Alguns deles são bastante simples, mas muito divertidos, e chegam mesmo a ser educativos.

É o caso do jogo de adivinhação chamado Maior ou Menor?, que vamos mostrar agora como programar, e que serve para ensinar estes conceitos às crianças.

O funcionamento do jogo é o seguinte:

O computador informa à criança que está pensando em um número situado entre 1 e 10. Em que número ela acha que ele está "pensando"?

Se a resposta que a criança der não for correta, o programa informa se o número é muito baixo ou muito alto, e pergunta novamente.

Quando a criança acertar, o programa informa quantas tentativas foram necessárias.

Este jogo é educativo também em outro sentido. Existe uma estratégia para acertar em menor número de vezes possível. Podemos deixar a criança jogar e descobrir sozinha qual é esta estratégia; ou então ensiná-la depois de certo tempo e ver se ela a aplica corretamente.

Depois veremos qual é esta estratégia!

Planejando o jogo

O programa terá as seguintes fases:

1. mostra na tela o título do jogo e as instruções iniciais;
2. sorteia um número aleatório entre 1 e 10;
3. pergunta um número para o jogador;
4. testa se este número é igual, menor ou maior do que o sorteado;
5. se for igual, o jogador acertou. Informa quantas tentativas precisou para acertar;
6. se for maior ou menor, informa através de uma mensagem, e volta para o passo 3. Incrementa o número de tentativas.

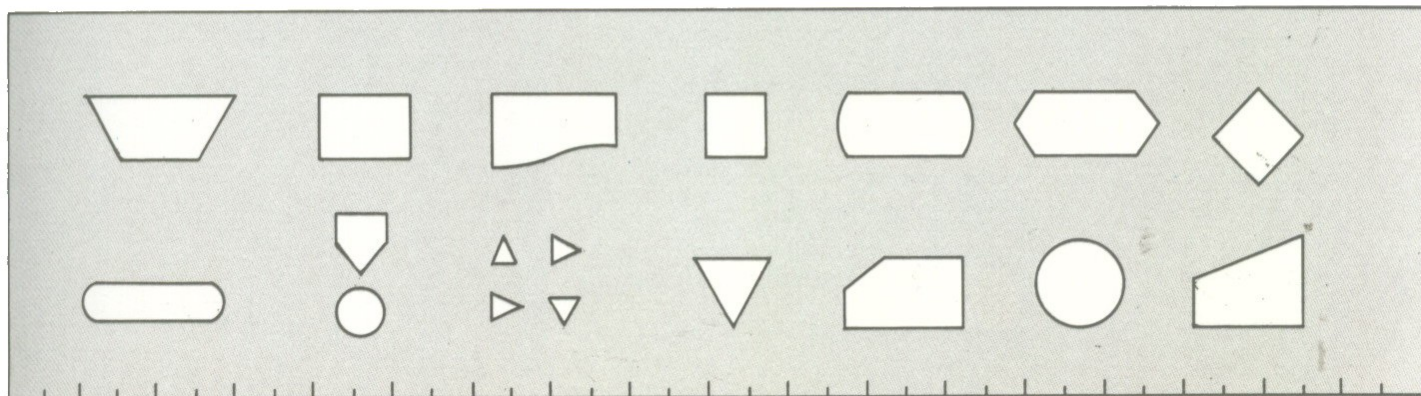
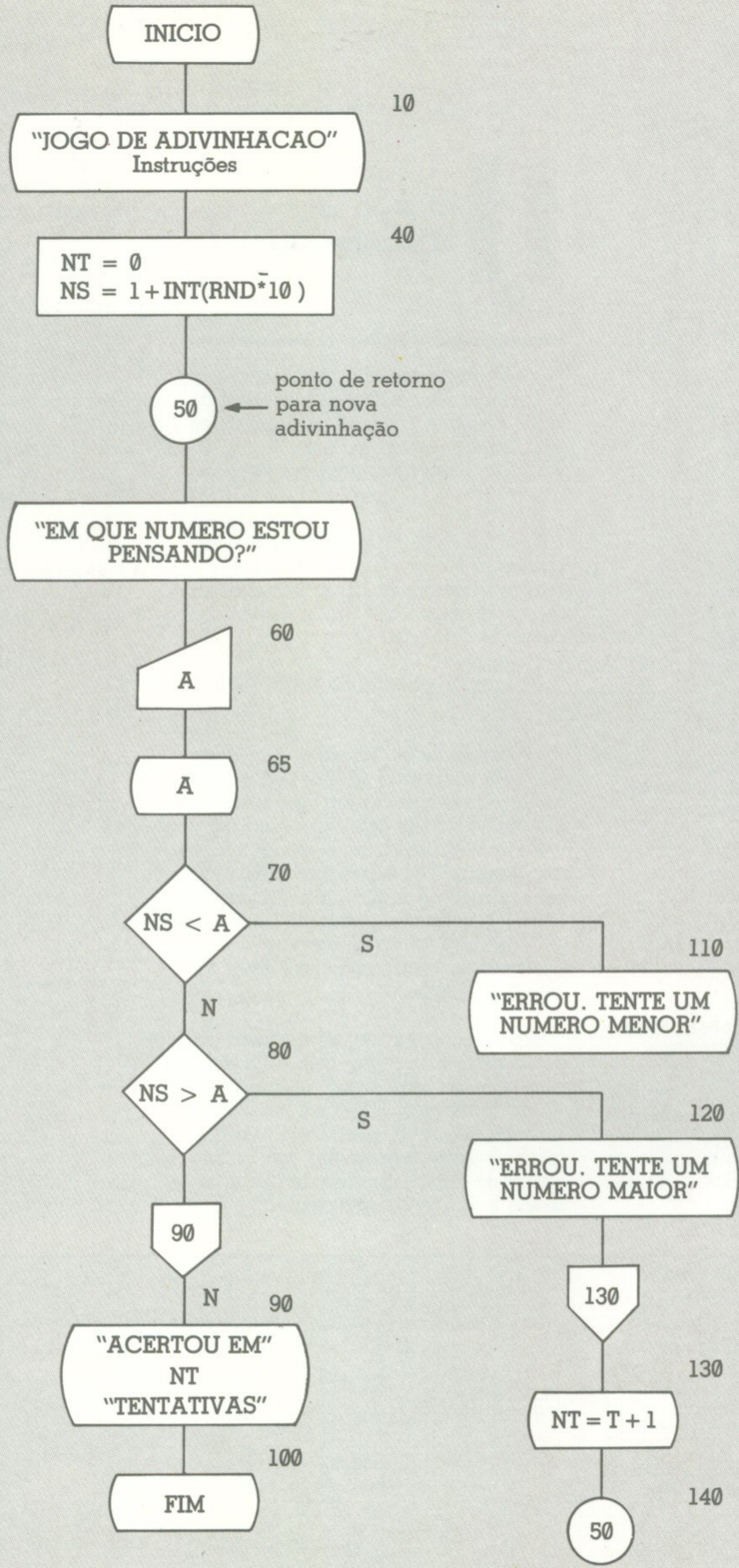
As variáveis do programa poderão ser as seguintes:

NS - número sorteado

A - adivinhação do jogador

NT - número de tentativas

Gabarito para desenho de fluxogramas

Fluxograma
INICIO
10
"JOGO DE ADIVINHACAO"
Instruções
40
NT = 0
NS = 1 + INT(RND*10)
ponto de retorno
para nova
adivinhação
50
"EM QUE NUMERO ESTOU
PENSANDO?"
60
A
65
A
70
NS < A
S
N
110
"ERROU. TENTE UM
NUMERO MENOR"
80
NS > A
S
120
"ERROU. TENTE UM
NUMERO MAIOR"
90
N
90
"ACERTOU EM"
NT
"TENTATIVAS"
130
130
NT = T + 1
100
FIM
140
50

Codificação em BASIC

```
1Ø PRINT "JOGO DE ADIVINHA-
   CAO"
15 PRINT " = = = = = = = =
   = = = = = = = = = = = ="
2Ø PRINT
25 PRINT "EU VOU PENSAR EM
   UM NUMERO DE 1 A"
3Ø PRINT "1Ø. GANHA QUEM
   ADIVINHAR EM MENOS"
35 PRINT "TENTATIVAS."
38 PRINT
4Ø LET NT=Ø
45 LET NS=1 + INT (RND*1Ø)
5Ø PRINT "EM QUE NUMERO ES-
   TOU PENSANDO?"
6Ø INPUT A
65 PRINT A
7Ø IF NS<A THEN GOTO 11Ø
8Ø IF NS>A THEN GOTO 12Ø
9Ø PRINT "ACERTOU EM ";NT;"
   TENTATIVAS"
1ØØ STOP
11Ø PRINT "ERROU. TENTE UM
    NUMERO MENOR"
115 GOTO 13Ø
12Ø PRINT "ERROU. TENTE UM
    NUMERO MAIOR"
13Ø LET NT=NT+1
14Ø GOTO 5Ø
```

```
JOGO DE ADIVINHACAO
= = = = = = = = = = = = = = = = = = = = = =

EU VOU PENSAR EM UM NUMERO DE 1 A 1Ø.
GANHA QUEM ADIVINHAR EM MENOS TENTA-
TIVAS.

EM QUE NUMERO ESTOU PENSANDO? 6. ER-
ROU. TENTE UM NUMERO MENOR. EM QUE NU-
MERO ESTOU PENSANDO? 3. ERROU. TENTE
UM NUMERO MAIOR. EM QUE NUMERO ESTOU
PENSANDO?

4 [L]
```

```
JOGO DE ADIVINHACAO
= = = = = = = = = = = = = = = = = = = = = =

EU VOU PENSAR EM UM NUMERO DE 1 A 1Ø.
GANHA QUEM ADIVINHAR EM MENOS TENTA-
TIVAS.

EM QUE NUMERO ESTOU PENSANDO? 6. ER-
ROU. TENTE UM NUMERO MENOR. EM QUE NU-
MERO ESTOU PENSANDO? 3. ERROU. TENTE
UM NUMERO MAIOR. EM QUE NUMERO ESTOU
PENSANDO? 4. ACERTOU EM 3 TENTATIVAS

9/1ØØ
```

Exemplo de um jogo

Coloque o programa no computador e experimente várias vezes.

Sugestões para modificar o programa

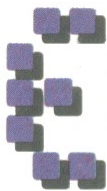
1. Coloque os números mínimos e máximos (no exemplo, 1 e 10) como variáveis dentro do programa, podendo ser especificadas pelos comandos INPUT, logo no início do programa.

2. Faça o jogador colocar o seu nome no início do programa e depois ponha este nome nas mensagens de erro e acerto.

3. Coloque no lugar do comando STOP uma pergunta:

QUER JOGAR MAIS UMA VEZ
(SIM OU NAO)?

4. Estenda o programa de modo a permitir a participação de dois jogadores, um competindo contra o outro (ganha quem adivinhar em menor número de tentativas).

5. Estenda o programa para adivinhação de letras do alfabeto (A, B, C, D etc.), ao invés de números.

Ensinando tabuada pelo computador

Uma boa aplicação para a função RND é a criação de um grande número de problemas de matemática, com dados diferentes de cada vez. Esta é uma aplicação educativa do computador, pois permite ao aluno fazer muitos exercícios. Além disso, cada aluno recebe um conjunto de exercícios totalmente diferente dos demais. Podemos também fazer o computador dar notas aos alunos, dependendo do número de problemas que ele acerta.

Vamos agora desenvolver juntos um programa para apresentar dez problemas de tabuada de multiplicação, sendo um dos fatores de multiplicação gerado ao acaso, entre 1 e 10. Após a apresentação de cada problema, a resposta é colhida e comparada com a resposta certa. No final é atribuída uma nota ao aluno.

Planejamento do programa

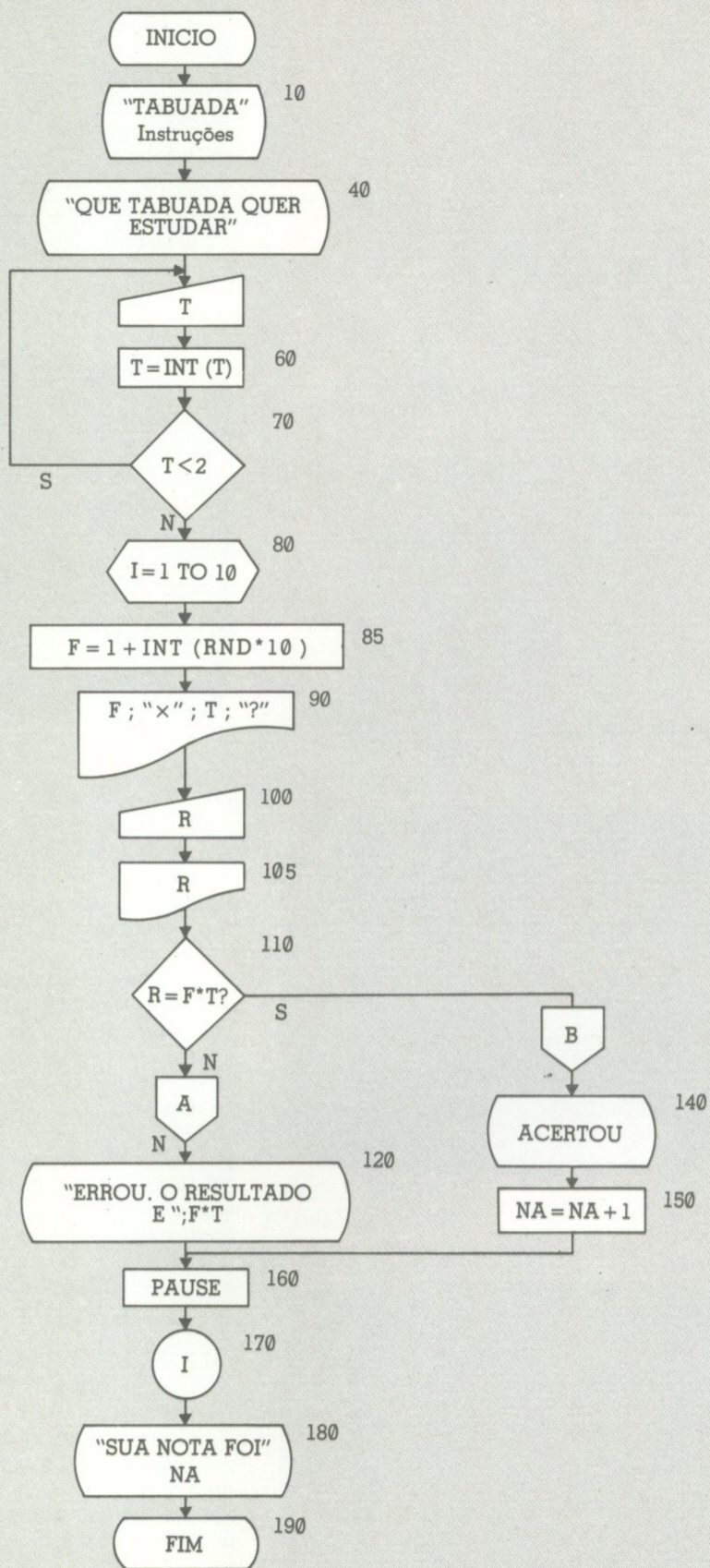
As fases do programa serão:

1. mostrar o título do programa e seu objetivo;
2. perguntar ao aluno que tabuada ele quer estudar;
3. sortear um número de 1 a 10;
4. mostrar na tela o problema: tabuada × número sorteado;
5. entrar a resposta do aluno;
6. testar se a resposta do aluno está correta;
7. se estiver correta, dar os parabéns e incrementar o contador de número de respostas corretas;
8. se estiver errada, mostrar o resultado correto;
9. voltar para o passo 3, até que se esgotem os dez problemas;
10. apresenta a nota final.

As variáveis poderão ser:

T - tabuada que quer estudar
NA - número de acertos
F - fator de multiplicação sorteado
R - resposta do aluno

Fluxograma

INICIO
10 "TABUADA" Instruções
40 "QUE TABUADA QUER ESTUDAR"
T
60 T=INT (T)
70 T<2 (S: volta para T; N: segue)
80 I=1 TO 10
85 F=1+INT (RND*10)
90 F ; "×" ; T ; "?"
100 R
105 R
110 R=F*T? (S: B; N: A)
A
120 "ERROU. O RESULTADO E ";F*T
B
140 ACERTOU
150 NA=NA+1
160 PAUSE
170 I
180 "SUA NOTA FOI" NA
190 FIM

Codificação em BASIC

```
10 PRINT "TABUADA"
15 PRINT "======="
20 PRINT
25 PRINT "VAMOS ESTUDAR TA-
   BUADA DE VEZES"
30 PRINT "EU APRESENTO UMA
   CONTA PARA VOCE"
35 PRINT "FAZER, E VOCE RES-
   PONDE, TA?"
37 PRINT
40 PRINT "QUE TABUADA QUER
   ESTUDAR?"
50 INPUT T
60 LET T=INT(T)
70 IF T<2 THEN GOTO 40
80 FOR I=1 TO 10
85 LET F=1 + INT (RND*10 )
90 CLS
95 PRINT F;"X";T;"?";
100 INPUT R
105 PRINT R
110 IF R=F*T THEN GOTO 140
120 PRINT "ERROU. O RESULTA-
    DO E ";F*T
130 GOTO 160
140 PRINT "ACERTOU"
150 LET NA=NA+1
160 PAUSE 200
170 NEXT I
180 PRINT "SUA NOTA FOI ";NA
190 STOP
```

Coloque o programa no computador e exercite com ele.

Sugestões para modificar o programa

1. Coloque o número de problemas (no exemplo, 10) como variável dentro do programa, podendo ser especificado pelo comando INPUT, logo no início do programa.

2. Faça o aluno colocar o seu nome no início do programa, e depois ponha este nome nas mensagens de erro e acerto.

3. Coloque no lugar do comando STOP uma pergunta:

QUER TENTAR MAIS UMA VEZ (SIM OU NAO)?

4. Estenda o programa de modo a fazer com que os dois fatores de multiplicação variem aleatoriamente.

5. No lugar de mensagem de acerto ou erro, coloque a figura gráfica de um palhacinho rindo ou chorando, respectivamente.

RESPOSTAS AOS EXERCÍCIOS ANTERIORES

Páginas 51/52

1. Faça um programa para calcular a soma dos números inteiros de 1 a 100.

Resposta:

Versão 1 (usando IF):

```
10 LET SOMA=1
20 LET N=1
30 LET N=N+1
40 IF N>100 THEN GOTO 70
50 LET SOMA=SOMA + N
60 GOTO 30
70 PRINT SOMA
```

Versão 2 (usando FOR):

```
10 LET SOMA=0
20 FOR N=1 TO 100
30 LET SOMA=SOMA + N
40 NEXT N
50 PRINT SOMA
```

2. Modifique o programa acima, de modo a poder calcular a soma dos números inteiros de 1 a um número qualquer.

Resposta:

```
 5 PRINT "ENTRE O NUMERO MA-
   XIMO PARA A SERIE:"
 8 INPUT MX
10 LET SOMA=0
20 FOR N=1 TO MX
30 LET SOMA = SOMA+N
40 NEXT N
50 PRINT SOMA
```

3. Faça um programa como o acima, para calcular o fatorial de um número inteiro qualquer (N!).

Resposta:

```
10 PRINT "QUE NUMERO QUER O
   FATORIAL?"
20 INPUT N
30 LET PROD=1
40 FOR I=2 TO N
50 LET PROD=PROD*N
60 NEXT I
50 PRINT "FATORIAL DE N=
   ";PROD
```

8. Faça um programa para aceitar um nome qualquer pelo teclado, e em seguida fazê-lo piscar dez vezes seguidas, no meio da primeira linha da tela.

Resposta:

```
10 PRINT "ENTRE UM NOME
   QUALQUER:"
20 INPUT N$
30 LET N=0
40 CLS
50 PRINT TAB (12);N$
60 PAUSE 30
70 LET N=N+1
80 IF N<10 THEN GOTO 40
90 STOP
```

9. Faça um programa para calcular a soma de dez números quaisquer, digitados pelo teclado.

Resposta:

```
10 LET SOMA=0
20 FOR I=1 TO 10
30 PRINT "ENTRE A PARCELA
   NO.";I,
40 INPUT P
45 PRINT P
50 LET SOMA=SOMA+P
60 NEXT I
70 PRINT SOMA
80 STOP
```

10. Faça um programa para imprimir uma tabela das raízes quadradas dos números de 10 a 1000, de 10 em 10.

Resposta:

```
10 PRINT "N", "RAIZ QUADRADA"
20 FOR N=10 TO 1000 STEP 10
30 PRINT N,SQR(N)
40 NEXT N
50 STOP
```

RESPOSTAS AO TESTE ANTERIOR

Página 69

9.17. Faça um programa para gerar na tela todas as combinações possíveis de preenchimento do cartão da Loteria Esportiva.

Solução:

```
10 LET N=0
20 FOR I=1 TO 3
30 FOR J=I TO 3
40 LET N=N+1
50 PRINT N;" - ";" COLUNA ";I
60 IF I=J THEN GOTO 80
70 PRINT " COLUNA ";J
80 NEXT J
90 NEXT I
```

Nota: Este programa gerará todas as combinações 1, 1-2, 1-3, 2, 2-3 e 3, mas não gerará a marcação tripla, 1-2-3.

9.20. Faça um programa para apresentar na tela três exercícios de tabuada de multiplicar e para testar se o resultado apresentado pelo aluno foi correto. O programa deve dizer, no fim, quantos exercícios foram resolvidos corretamente pelo aluno.

Solução:

```
10  PRINT "3 EXERCICIOS DE TA-
    BUADA"
20  PRINT
25  LET N=0
30  PRINT "3×4?"
40  INPUT R
50  IF R<>12 THEN GOTO 90
60  LET N=N+1
70  PRINT "ACERTOU"
80  GOTO 100
90  PRINT "ERROU"
100 PRINT "8×6?"
120 INPUT R
130 IF R<>48 THEN GOTO 170
140 LET N=N+1
150 PRINT "ACERTOU"
160 GOTO 180
170 PRINT "ERROU"
180 PRINT "7×4?"
190 INPUT R
200 IF R<>28 THEN GOTO 240
210 LET N=N+1
220 PRINT "ACERTOU"
230 GOTO 250
240 PRINT "ERROU"
250 PRINT
260 PRINT "VOCE ACERTOU
    ";N;" PROBLEMAS"
270 STOP
```

Observação: Note como esta é uma maneira pouco eficiente de fazer um programa que apresenta programas repetitivamente. Um programa muito mais curto seria:

```
10  PRINT "PROBLEMAS DE TA-
    BUADA"
20  PRINT
50  PRINT "QUE TABUADA?"
60  INPUT T
65  LET N=0
70  FOR I=1 TO 10
80  CLS
90  PRINT T;"X ";I;" ? "
100 INPUT R
110 IF R<>I*T THEN GOTO 130
120 LET N=N+1
125 GOTO 140
130 PRINT "ERRADO"
140 NEXT I
150 PRINT
160 PRINT VOCE ACERTOU ";N;"
    PROBLEMAS
170 STOP
```

RESPOSTAS AOS EXERCÍCIOS ANTERIORES

Páginas 62/63

1. Faça um programa para simular várias vezes o sorteio de uma moeda.

Resposta:

```
10  PRINT "SORTEIO DE UMA
    MOEDA"
15  RAND
20  PRINT
30  PRINT "QUANTAS VEZES?"
40  INPUT N
50  FOR I=1 TO N
60  IF RND>0.5 THEN 90
70  PRINT "CARA"
80  GOTO 100
90  PRINT "COROA"
100 NEXT I
```

2. Modifique o programa acima para contar quantas vezes saiu cara e quantas vezes saiu coroa, informando o resultado no fim.

Resposta:

```
10  PRINT "SORTEIO DE UMA
    MOEDA"
15  RAND
20  PRINT
30  PRINT "QUANTAS VEZES?"
40  INPUT N
42  LET CA=0
44  LET CO=0
50  FOR I=1 TO N
60  IF RND>0.5 THEN 90
70  PRINT "CARA"
75  LET CA=CA+1
80  GOTO 100
90  PRINT "COROA"
95  LET CO=CO+1
100 NEXT I
110 CLS
120 PRINT "SORTEIO DE UMA
    MOEDA ";N;" VEZES"
130 PRINT
140 PRINT "SAIRAM ";CA;" CA-
    RAS (";CA/N*100 ;" %)"
150 PRINT "SAIRAM";CO;" CO-
    ROAS (";CO/N*100 ;" %)"
160 STOP
```

5. Faça um programa para sortear as cartas de um baralho, mostrando o seu valor ou figura na tela.

Resposta:

```
10  REM --- SORTEIO DE UM BA-
    RALHO
20  REM --- PARA MICROS TIPO
```

```
    SINCLAIR ZX81
30  REM --- JANEIRO 85 - (C) R.
    SABBATINI
40  REM --------------------------------
50  REM
54  RAND
56  REM
60  REM --- TITULO E DADOS
70  REM
80  PRINT "JOGO DE SORTEIO
    DE CARTAS"
90  PRINT "============
    ============="
100 PRINT
110 PRINT "QUANTAS CARTAS
    QUER SORTEAR?"
120 INPUT NC
130 LET NC=INT (NC)
140 IF NC<1 THEN GOTO 110
150 REM
160 REM --- ALCA DE SORTEIO
    DAS CARTAS
170 REM
180 FOR I=1 TO NC
190 LET C=2 + INT (RND * 11)
200 LET N=1 + INT (RND * 4)
210 REM
220 REM --- MOSTRA O RESULTA-
    DO
230 REM
240 PRINT "A CARTA SORTEADA
    FOI";
250 IF C<11 THEN GOTO 310
252 REM
254 REM --- MOSTRA VALOR DA
    CARTA
256 REM
260 IF C=11 THEN PRINT "VALE-
    TE";
270 IF C=12 THEN PRINT
    "DAMA"
280 IF C=13 THEN PRINT "REI";
290 IF C=14 THEN PRINT "AS";
300 GOTO 330
310 PRINT C;
330 PRINT " DE ";
340 REM
350 REM --- IMPRIME NAIPE
360 REM
370 IF N=1 THEN PRINT "ESPA-
    DAS"
380 IF N=2 THEN PRINT "COPAS"
390 IF N=3 THEN PRINT "PAUS"
400 IF N=4 THEN PRINT
    "OUROS"
410 NEXT I
420 STOP
```

6. Faça um programa para imprimir 20 sorteios de um número aleatório R, calculando e imprimindo em cada sorteio as seguintes transformações do programa: R * 10, 1+INT(R*10), 10 +INT(R*10)

Resposta:

```
10 RAND
20 FOR I=1 TO 20
30 LET R=RND
40 PRINT R*10 ;1+INT(R*10 );
   10 +INT(R*10 )
50 NEXT I
60 STOP
```

Página 64

2. Faça um programa que sorteie dois dados, e mostre a soma dos números sorteados. Se a soma for igual a 7, pare de sortear. Se for diferente de 7, sorteie novamente.

Resposta:

```
10 PRINT "O JOGO DOS 2 DA-
   DOS"
20 PRINT
30 RAND
40 LET D1=1+INT(RND*6)
50 LET D2=1+INT(RND*6)
60 LET SOMA=D1+D2
70 IF SOMA=7 THEN STOP
80 GOTO 40
```

3. Faça um programa que sorteie as cartas de um baralho completo, perguntando antes ao jogador qual naipe ele acha que sairá sorteado.

Resposta:

Vamos aproveitar o programa do exercício 5, mostrado anteriormente.

As linhas novas são: 182, 183, 184, 402 e 404

```
10  REM --- SORTEIO DE UM BA-
    RALHO
20  REM --- PARA MICROS TIPO
    SINCLAIR ZX81
30  REM --- JANEIRO 85 - (C) R.
    SABBATINI
40  REM --------------------------------
50  REM
54  RAND
56  REM
60  REM --- TITULO E DADOS
70  REM
80  PRINT "JOGO DE SORTEIO
    DE CARTAS"
90  PRINT "============
    ============="
100 PRINT
110 PRINT "QUANTAS CARTAS
    QUER SORTEAR?"
120 INPUT NC
130 LET NC=INT (NC)
140 IF NC<1 THEN GOTO 110
150 REM
160 REM --- ALCA DE SORTEIO
    DAS CARTAS
170 REM
180 FOR I=1 TO NC
182 PRINT "QUE NAIPE VAI DAR?
183 PRINT "1=ESPADAS, 2=CO-
    PAS, 3=PAUS, 4=OUROS"
184 INPUT A
190 LET C=2 + INT (RND * 11)
200 LET N=1 + INT (RND * 4)
210 REM
220 REM --- MOSTRA O RESULTA-
    DO
230 REM
240 PRINT "A CARTA SORTEADA
    FOI";
250 IF C<11 THEN GOTO 310
252 REM
254 REM --- MOSTRA VALOR DA
    CARTA
266 REM
260 IF C=11 THEN PRINT "VALE-
    TE";
270 IF C=12 THEN PRINT
    "DAMA";
280 IF C=13 THEN PRINT "REI";
290 IF C=14 THEN PRINT "AS";
300 GOTO 330
310 PRINT C;
330 PRINT " DE ";
340 REM
350 REM --- IMPRIME NAIPE
360 REM
370 IF N=1 THEN PRINT "ESPA-
    DAS"
380 IF N=2 THEN PRINT "COPAS"
390 IF N=3 THEN PRINT "PAUS"
400 IF N=4 THEN PRINT
    "OUROS"
402 IF N=A THEN PRINT "ACER-
    TOU"
404 IF N <> A THEN PRINT "ER-
    ROU"
410 NEXT I
```

5. Faça um programa para simular o seguinte jogo ("Porquinho"). Veja o enunciado completo.

```
10  REM --- JOGO DO PORQUI-
    NHO
20  REM ---------------------------
30  REM
35  PRINT "VAI ROLAR O
    DADO..."
37  PRINT
40  RAND
50  LET SOMA=0
60  LET D=1+INT(RND*6)
65  PRINT "SAIU O NUMERO ";D
70  IF D=1 THEN GOTO 160
80  LET SOMA=SOMA+D
90  PRINT "VOCE AGORA TEM
    ";SOMA;" PONTOS."
100 PRINT
110 PRINT "QUER CONTINUAR
    (SIM OU NAO)?"
120 INPUT R$
130 IF R$="SIM" THEN GOTO 60
140 PRINT "COVARDE..."
150 STOP
160 PRINT "PERDEU TUDO..."
170 STOP
```

S TABELAS DO COMPUTADOR

Até agora, você já viu como podemos armazenar valores na memória do computador, pelo uso das *variáveis*.

Rememorando, uma variável é um nome simbólico, dado pelo programador a uma ou mais memórias do computador, de modo a indicar onde será armazenado um determinado valor. Até agora, aprendemos que existem em BASIC dois tipos fundamentais de variáveis:

- as *variáveis numéricas*
- as *variáveis literais* (ou *alfanuméricas*)

As variáveis numéricas servem apenas para armazenar números. Por exemplo:

```
5 LET NUM = 12.6

15 LET MEDIA = (N1 + N2)/2
```

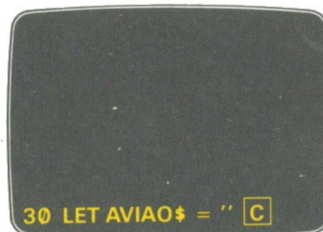
No primeiro exemplo, estamos dando ao computador uma instrução para armazenar o valor numérico 12.6 na variável que chamamos de NUM.

No segundo exemplo, estamos dando uma instrução para armazenar na variável chamada MEDIA, o resultado da expressão aritmética à direita do sinal de igualdade (que por sua vez está usando o conteúdo de duas outras memórias, ou seja, N1 e N2).

As variáveis literais, por sua vez, servem para armazenar palavras frases, etc., ou qualquer outra seqüência de símbolos que o computador consiga gerar. Por exemplo:

```
12 LET NOME$ = "TANCREDO
   NEVES"

15 LET ESPACO$ = "          "

20 LET XINGA$ = "$%..&...?& ??"

30 LET AVIAO$ = "▄▂▄▂"
```

Observe que é necessário colocar o sinal de cifrão ($) no final do nome de uma variável literal, o que serve para "avisar" ao computador que aquela variável deverá ser usada para armazenar caracteres e não valores numéricos. (Por extensão, dizemos que esse tipo de variável serve para armazenar *valores literais*, embora eles não sejam exatamente valores.)

Além disso, podemos ver, pelo terceiro exemplo, que é possível armazenar caracteres gráficos dentro de uma variável literal. Para conseguir isto no TK 85, basta pressionar as teclas SHIFT e 9, logo depois de termos digitado o primeiro sinal de aspas:

Esta combinação de teclas causa o aparecimento do cursor G na linha de entrada, o que possibilita a digitação de caracteres gráficos, a partir daí:

Como mostramos a seguir:

Depois de digitar os gráficos, devemos pressionar de novo as teclas SHIFT e 9, para podermos teclar o sinal de "fecha aspas".

Você deve se lembrar também de que o nome de uma variável literal pode representar mais do que uma "caixinha" na memória do computador (reveja o esquema da página 27). Na realidade, cada letra ou caractere entre as aspas irá ocupar uma memória após a outra. O conjunto de memórias usado para armazenar todos os caracteres de um valor literal recebe o nome técnico de *cordão*, neste caso (o nome correspondente em inglês, *string*, também é muito usado).

m novo tipo de conjunto

Neste capítulo, vamos conhecer um novo tipo de conjunto. Com ele, podemos utilizar um nome único de variável, para armazenar vários valores separados. Eles são chamados de *conjuntos indexados* e são muito úteis para se colocar na memória do computador coisas organizadas na forma de *tabelas*.

Para entender melhor como funciona um conjunto desse tipo, vamos "bolar" um programa para construir uma tabela dos pontos ganhos pelos vários times de futebol em um campeonato.

Uma maneira de realizar isto com as variáveis "normais" que você aprendeu até agora seria:

```
10 PRINT "CAMPEONATO BRASI-
   LEIRO DE FUTEBOL"
20 PRINT" TABELA DE PONTOS
   GANHOS"
30 PRINT
40 REM
50 REM === ENTRADA DE
   DADOS
60 REM
70 PRINT "ENTRE O NUMERO DE
```

```
   PONTOS GANHOS"
75 PRINT "PARA CADA TIME DA
   TABELA:"
80 PRINT
90 PRINT "TIME NUMERO 1 ? ";
100 INPUT T1
110 PRINT T1
120 PRINT "TIME NUMERO 2 ? ";
130 INPUT T2
140 PRINT T2
```

e assim por diante. Teríamos que colocar, em seguida, várias linhas semelhantes, para entrar os valores dos times T3, T4, T5 etc., até incluir todos os times (quantos existissem!).

Em seguida, poderíamos escrever um outro pedaço do programa, para mostrar o número médio de pontos do campeonato, por exemplo:

```
400 LET MEDIA = (T1 + T2 + T3
    + T4 + T5)/5
410 PRINT
420 PRINT "MEDIA DOS PONTOS
    GANHOS = ";MEDIA
430 STOP
```

Você já percebeu que esse método de programação é muito trabalhoso, e muito limitado:

— tenho que definir de antemão o número de times que vão figurar na tabela. Se este número mudar depois, terei que modificar o programa em vários lugares (inclusive na fórmula da linha 400);

— quanto maior for o número de times, mais comprido será o programa, e mais trabalhoso será escrevê-lo.

Por isso, seria ideal se pudéssemos usar um único nome para as variáveis T1, T2, T3 etc. Assim, poderíamos guardar os diversos valores numéricos em "caixinhas" ligadas umas nas outras, mais ou menos como no caso das variáveis literais.

Felizmente, existe esse tipo de *variável conjunta* na linguagem BASIC. Com ela, podemos utilizar um único nome (por exemplo, T) para a *área de memória* que armazenará os pontos ganhos de cada equipe:

	área de memória				
variável T	12	10	9	5	3
número da memória	1	2	3	4	5

Para colocar um valor nesse tipo de variável, podemos usar um comando LET ou comando INPUT:

```
100 INPUT T(1)
```

O valor entre parênteses, que vem depois do nome do conjunto, é chamado de *índice* ou *subscrito*. Ele indica qual é a "caixinha" onde será colocado o valor digitado pelo teclado. Neste exemplo, a memória indicada é a 1, ou seja, o número de pontos ganhos será armazenado para o primeiro time. Outro exemplo:

```
55 LET T(5) = 3
```

Aqui, estamos dando a seguinte instrução ao computador:

PEGUE o número 3 e ARMAZENE na memória número 5 da variável chamada T.

Da mesma forma que os comandos LET e INPUT, podemos usar uma variável indexada dentro de outras instruções, como o PRINT. Por exemplo, se dermos a instrução:

```
56 PRINT T (5)
```

estamos querendo dizer:

PEGUE o valor contido na "caixinha" número 5 da variável T e mostre na tela.

Ou ainda:

```
57 LET SOMA = T(1) + T(2)
```

o que quer dizer:

SOME os valores contidos nas memórias 1 e 2 da variável T, e armazene o resultado na variável chamada SOMA.

As memórias chamadas T(1), T(2), T(5) etc., neste exemplo, recebem o nome de variáveis *indexadas*, porque são assinaladas por um *índice*, ou seja, o número de ordem delas, no *conjunto*.

As memórias T(1), T(2) etc. são chamadas também de *elementos* de T.

Também são muito usados estes nomes: variáveis subscritas (indexadas) e subscritos (índices).

Quando queremos ler em voz alta (pronunciar) o nome de um elemento subscritado, falamos assim:

B(5) bê índice 5 *ou* bê sub 5

G(N) gê índice ene *ou* gê sub ene

Portanto, um conjunto ocupa na memória uma série de "caixinhas" individuais, cujo "endereço" consiste num nome comum, seguido do subscrito, entre parênteses:

endereço	conteúdo
T(1)	23
T(2)	12
T(3)	9
T(4)	7
T(5)	4

A grande vantagem desse tipo de variável é que os *índices* dos elementos também podem ser variáveis.

Veja este exemplo:

```
1000 LET V(1) = 456
1010 LET V(2) = 132
1020 LET V(3) = 111
1030 PRINT
1040 PRINT "QUE VALOR QUER
     VER (1, 2 OU 3)? ";
1050 INPUT N
1060 PRINT N
1070 PRINT "VALOR = ";V(N)
1080 GOTO 1030
```

— As linhas 1000, 1010 e 1020 servem para colocar valores numéricos dentro dos elementos 1, 2 e 3 do conjunto V.

— As linhas 1040 e 1050 servem para perguntar ao usuário qual é o número de ordem do elemento que ele quer ver na tela (este número é armazenado na variável N).

— A linha 1060 serve para mostrar na tela o conteúdo do elemento N do conjunto V!

Eis como ficaria um exemplo de utilização do programa:

Primeiro vamos examinar o conteúdo do segundo elemento do conjunto (para isto digitamos o valor 2, quando aparece a pergunta na tela):

O programa mostra o valor pedido, em seguida repete a pergunta. Agora queremos examinar o elemento número 1:

E assim por diante:

```
QUE VALOR QUER VER
(1, 2 OU 3)? 2  VALOR = 132
QUE VALOR QUER VER
(1, 2 OU 3)? 1 VALOR = 456
QUE VALOR QUER VER
(1, 2 OU 3)?

1 C
```

Tornando o programa mais eficiente

Vamos usar agora o nosso conhecimento a respeito das variáveis indexadas, para modificar o programa da tabela de pontos do campeonato. Inicialmente, vamos fazer um programa para cinco times apenas:

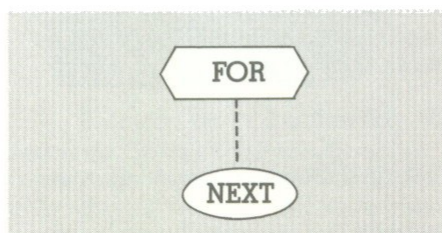
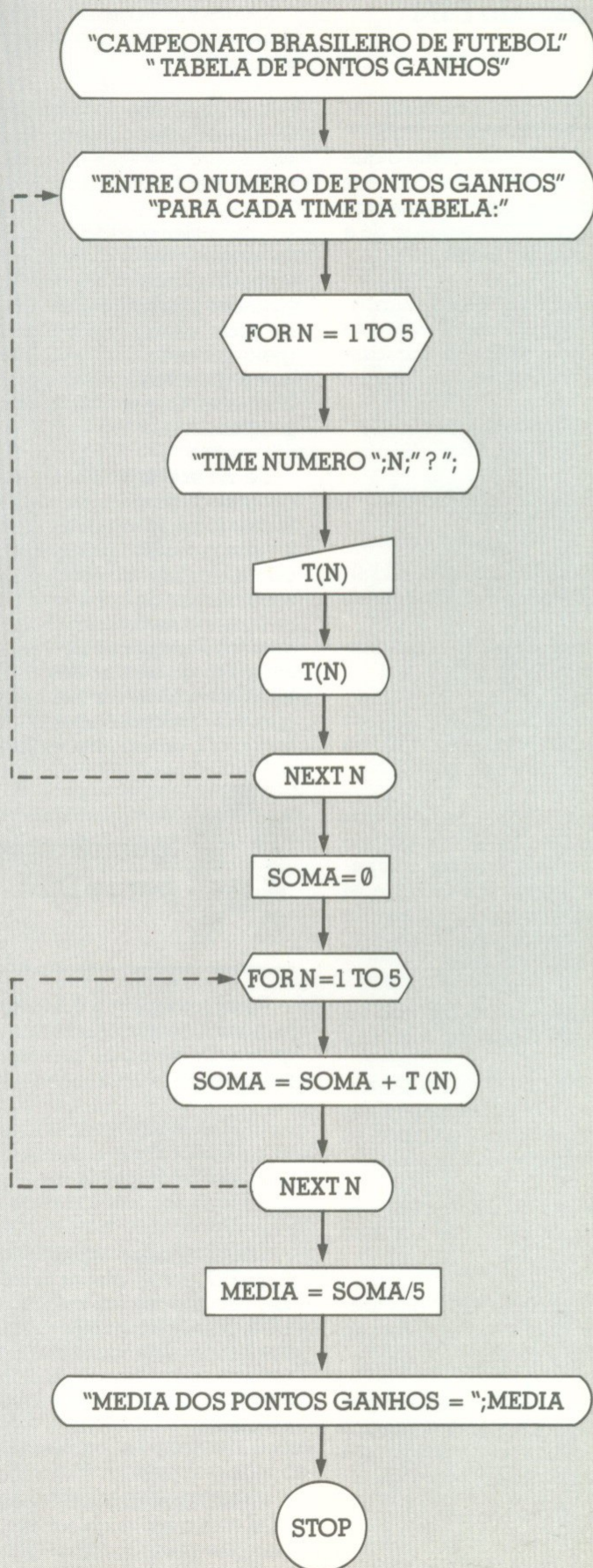
```
10 PRINT "CAMPEONATO
   BRASILEIRO DE FUTEBOL"
20  PRINT " TABELA DE PONTOS
   GANHOS"
30 PRINT
40 REM
50 REM = = = ENTRADA DE
   DADOS
60 REM
70 PRINT "ENTRE O NUMERO DE
   PONTOS GANHOS"
75 PRINT "PARA CADA TIME DA
   TABELA:"
80 PRINT
85 FOR N = 1 TO 5
90 PRINT "TIME NUMERO ";N;" ? ";
100 INPUT T(N)
110 PRINT T(N)
120 NEXT N
130 REM
140 REM = = = CALCULO DA
    MEDIA
150 REM
160 LET SOMA = 0
170 FOR N = 1 TO 5
180 LET SOMA = SOMA + T(N)
190 NEXT N
200 REM
210 REM = = = MOSTRA
    RESULTADOS
220 REM
400 LET MEDIA = SOMA/5
410 PRINT
420 PRINT "MEDIA DOS PONTOS
    GANHOS = ";MEDIA
430 STOP
```

Eis aqui um exemplo de utilização do programa:

```
CAMPEONATO BRASILEIRO DE
FUTEBOL
TABELA DE PONTOS GANHOS

ENTRE O NUMERO DE PONTOS
GANHOS PARA CADA TIME DA
TABELA:

TIME NUMERO 1 ? 23
TIME NUMERO 2 ? 12
TIME NUMERO 3 ?  9
TIME NUMERO 4 ?  7
TIME NUMERO 5 ?  4

MEDIA DOS PONTOS
GANHOS = 11
```

Vamos entender agora como funciona o programa.

Para isto, vamos fazer o oposto do que foi ensinado para você. Vamos fazer agora o fluxograma do programa (normalmente ele deve ser feito antes deste ser escrito, lembra-se?).

Para representar uma *alça* do tipo FOR ... NEXT, usamos os blocos abaixo:

Desse modo, o programa ficaria assim:

A primeira alça, que vai da linha 85 a 12Ø, serve para entrarmos sucessivamente os times de 1 a 5. A variável N é usada como contadora da alça e, ao mesmo tempo, como índice do conjunto T.

Esta é uma técnica muito usada na programação de conjuntos, e se chama *indexação indireta*.

Você pode ver que, a cada valor de N incrementado pelo FOR, o programa referencia um novo valor de T nas linhas 1ØØ (INPUT) e 11Ø (PRINT). Ao mesmo tempo, serve também para indicar ao usuário qual o número do time para o qual ele deve digitar os pontos ganhos.

Valor de N	Elemento referenciado	Valor digitado (exemplo)
1	T (1)	23
2	T (2)	12
3	T (3)	9
4	T (4)	7
5	T (5)	4

A segunda alça, que vai das linhas 17Ø a 19Ø, serve para calcular a soma dos pontos ganhos por todos os times. Esse valor será usado no cálculo da média, dividindo-o depois por 5. Para servir como acumuladora, usamos a variável SOMA, que deve ser zerada antes da entrada na alça (linha 16Ø).

Portanto, como você viu, o programa ficou muito mais bem escrito, conciso e eficiente.

Podemos perceber mais facilmente a utilidade do conjunto indexado se tivermos que mudar o número de times. Por exemplo, se fossem dez times em vez de cinco, teríamos que mudar apenas as linhas 85, 17Ø e 4ØØ:

```
85 FOR N = 1 TO 1Ø

17Ø FOR N = 1 TO 1Ø

4ØØ LET MEDIA = SOMA/1Ø
```

Você já percebeu que seria muito mais fácil colocar o número de times como uma variável NT, que seria perguntada ao usuário logo no início do programa, assim:

```
52 PRINT "QUANTOS TIMES ?"
54 INPUT NT
```

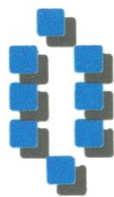
Assim, o programa fica muito mais *genérico*, ou seja, pode ser usado em qualquer situação, sem necessidade de se mudar o programa a cada momento.

Outra modificação interessante que podemos fazer ainda é mostrar de novo, em uma tela limpa, os valores digitados, antes da média. Isto é possível porque eles estão todos armazenados no conjunto T.

Portanto, o nosso programa modificado fica assim agora:

```
10 PRINT "CAMPEONATO
   BRASILEIRO DE FUTEBOL"
20 PRINT " TABELA DE PONTOS
   GANHOS"
25 PRINT "       VERSAO 2"
30 PRINT
40 REM
50 REM === ENTRADA DE
   DADOS
51 REM
52 PRINT "QUANTOS TIMES ?"
54 INPUT NT
70 PRINT "ENTRE O NUMERO DE
   PONTOS GANHOS"
75 PRINT "PARA CADA TIME DA
   TABELA:"
80 PRINT
85 FOR N = 1 TO NT
90 PRINT "TIME NUMERO ";N;" ? ";
100 INPUT T(N)
110 PRINT T(N)
120 NEXT N
130 REM
140 REM === CALCULO DA
    MEDIA
150 REM
160 LET SOMA = 0
165 CLS
170 FOR N=1 TO NT
175 PRINT "TIME NUMERO ";N;" =
    ";T(N); "PONTOS GANHOS"
180 LET SOMA = SOMA + T(N)
190 NEXT N
200 REM
210 REM === MOSTRA
    RESULTADOS
220 REM
400 LET MEDIA = SOMA/NT
410 PRINT
420 PRINT "MEDIA DOS PONTOS
    GANHOS = ";MEDIA
430 STOP
```

A tela com os resultados ficaria assim agora:

```
TIME Nº 1 = 23 PONTOS GANHOS
TIME Nº 2 = 12 PONTOS GANHOS
TIME Nº 3 = 9 PONTOS GANHOS
TIME Nº 4 = 7 PONTOS GANHOS
TIME Nº 5 = 5 PONTOS GANHOS

MEDIA DOS PONTOS
GANHOS = 11
```


O comando DIM

Um *conjunto indexado*, como o que você viu nos exemplos acima, pode ser considerado também como se fosse uma *lista*. O outro nome técnico que se dá a esse tipo de conjunto é *vetor*.

Para diferenciar uma variável numérica, que só pode armazenar um valor de cada vez, de uma variável conjunta, usamos a seguinte nomenclatura:

variável numérica simples =
ESCALAR

variável numérica conjunta =
VETOR

A nomenclatura acima é tirada da álgebra linear (uma parte da matemática).

Quantos elementos pode ter um conjunto na memória?

Existem algumas limitações quanto ao emprego desses elementos no BASIC do computador que você está usando.

Inicialmente, um conjunto numérico pode ter no máximo, dez elementos, o que dá para algumas aplicações, mas para outras é muito pouco. Por exemplo, se quiséssemos colocar todos os times da Copa Brasil, seria muito mais do que dez, e o computador não aceitaria.

Entretanto, existe um jeito de colocar mais de dez elementos em um conjunto.

Para isto, temos que *declarar* previamente ao computador quantas memórias ele tem que reservar para o conjunto, se ele for conter mais de dez elementos.

O comando pelo qual se faz essa declaração chama-se DIM (que significa DIMensão).

Dimensionamento é o termo que usamos para dizer quantos elementos, no máximo, conterá o conjunto.

No exemplo dos times, se quiséssemos colocar um máximo de setenta times, por exemplo (N = 70), seríamos obrigados a colocar no programa a seguinte linha:

```
45 DIM T(70)
```

Esta declaração está dizendo ao computador:

RESERVE uma dimensão de setenta elementos para o conjunto chamado T.

Outras limitações de memória para o dimensionamento dos conjuntos são as seguintes:

- um conjunto não pode ter mais de 65 535 elementos. Se este número for ultrapassado por engano, o computador indicará a mensagem de erro tipo B;

- cada elemento de um conjunto ocupa quatro *bytes*. A soma dos números de todos os elementos de conjuntos dimensionados implicitamente (isto é, menos de onze elementos) e explicitamente (pelo comando DIM) não deve ser superior à quantidade de memória central disponível (o que está sobrando, fora o programa).

Se, na execução de um programa, a quantidade de memória não for suficiente para todas as variáveis e conjuntos definidos, o computador indicará um erro tipo 4. Quando isso acontece, muitas vezes o número da linha onde ocorre o erro (que, na mensagem de erro na tela, aparece depois do sinal /) pode estar incompleto ou errado, devido também à falta de memória. Por exemplo, a mensagem 4/200 poderá aparecer incompleta, assim: 4/2, dando uma indicação falsa.

Algumas regras para o DIM

Podemos repetir o comando DIM para quantos conjuntos houver no meu programa. Deve-se usar um comando DIM para cada conjunto.

```
10 DIM A(65)
15 DIM B(12)
20 DIM FG(8)
```

Observe, nesse exemplo, que o conjunto FG, por ter menos de oito elementos, não precisa ser colocado na linha DIM. Entretanto, muitos programadores têm hábito (aliás, muito salutar) de declarar no começo do programa todas as variáveis indexados, para fins de documentação.

Outra regra importante para o uso do DIM é que ele pode ser colocado *em qualquer ponto de um programa*, porém sempre *antes de ser referenciado* o *conjunto pela primeira vez*.

Assim, estaria errado o seguinte programa:

```
10 FOR I=1 TO 100
20 INPUT T(I)
30 NEXT I
40 DIM T(100)
```

O programa só iria indicar o erro quando, durante a execução, a variável contadora I atingisse o número 11. Isso é explicado pelo fato de que o interpretador BASIC "garante" a dimensão de um conjunto, mesmo que não declarado em DIM, até dez elementos! O erro indicado pelo computador será de tipo 3 (subscrito fora da faixa).

Em conseqüência, outra regra muito importante para o uso correto do DIM é que não se deve dimensionar um conjunto com um número menor do que o valor dimensionado. Por exemplo:

```
 5 DIM T(50)
10 FOR I=1 TO 100
20 LET T(I)=0
30 NEXT I
```

O programa iria executar normalmente, até atingir I = 51, quando então o índice na linha 20 seria maior do que a capacidade do conjunto T.

Os conjuntos numéricos definidos em BASIC têm a seguinte característica importante: existe sempre um elemento numerado com o zero. Por exemplo: T(Ø) é um elemento válido do conjunto dimensionado T:

```
DIM T(5)
```

Esse elemento pode ser usado por comandos INPUT, LET, PRINT etc., sem problemas. Por isso, o conjunto T dimensionado como no exemplo acima, na realidade tem *seis* elementos do comando DIM disponíveis, e não *cinco*, como poderia parecer. Você pode decidir usá-lo ou não, dependendo da finalidade do programa.

Finalmente, outra particularidade da declaração DIM.

Nos computadores com a lógica Sinclair, isto é, o TK83, o TK85, o CP200 etc., podemos *redeclarar* um conjunto, ou seja, mudar a sua dimensão em pontos diferentes do programa. Por exemplo:

```
10 DIM A(10)
20 FOR I=TO 10
30 LET A(I)=I
40 NEXT I
50 DIM A(5)
.
.
.
etc.
```

Neste caso, o conjunto A inicial é cancelado da memória (e com ele os dados armazenados), e é dimensionado um conjunto A com apenas cinco elementos.

É preciso observar entretanto, que isso não é permitido na linguagem BASIC padrão e em muitos outros computadores. Nesses outros computadores é necessário dar antes o comando ERASE ou CLEAR, dependendo da marca, para "limpar" a memória para outro conjunto com o mesmo nome.

EXERCÍCIOS

1. Indique abaixo se uma variável é indexada do tipo vetor (V), indexada do tipo matriz (M) ou escalar (E)

A(3)--- A3--- X(5) --- ZZZ---T(12,14)---
N(7)--- I --- I(2,2)--- I22 ---VETOR ---

2. Modifique o programa de cálculo dos pontos ganhos dos times, de modo que mostre na tela de resultados qual a diferença entre o número de pontos ganhos por cada time e a média geral. Por exemplo:

```
TIME NUMERO 1 = 23 PONTOS
GANHOS. DIFERENCA DA ME-
DIA = 12 PONTOS
TIME NUMERO 2 = 12 PONTOS
GANHOS. DIFERENCA DA ME-
DIA = 1 PONTO
.
.
.
etc.
```

3. Seja o conjunto C, com cinco elementos. Preencha os elementos de A com os valores abaixo:

3.1. C(Ø) = 3
C(1) = 12
C(3) = 78
C(4) = C(1)+C(3)/2
C(5) = C(1)+2

E agora escreva abaixo o que cada um dos programas seguintes vai mostrar na tela:

3.2.
```
10 FOR I=Ø TO 5
20 PRINT C(I);
30 NEXT I
```

3.3.
```
10 LET N=3
20 PRINT C(1)+C(N)-C(N+2)
30 PRINT N*2 + C(N-3)
```

3.4.
```
10 LET C(5)=C(C(Ø))
20 PRINT C(5);C(Ø)
```

4. Considere que os conjuntos A e B e as variáveis escalares X, Y e Z abaixo receberam os seguintes valores:

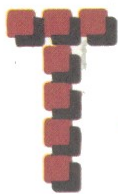
A(1)	5	B(Ø)	3	X	Ø
A(2)	25	B((1)	123	Y	1
A(3)	−9	B(2)	25.5	Z	2

E indique agora quais serão os valores amostrados pelos comandos abaixo:

```
PRINT A(1) ----
PRINT Z    ----

PRINT B(Z) ---
PRINT A(A(1)-Z)---

PRINT A(B(Ø)) ----
PRINT B(Y)-Y ----
```

5. Indique o que cada programa abaixo mostrará na tela:

5.1.
```
10 DIM VET(8)
20 FOR I=Ø TO 8
30 LET VET (I) =I+1
40 NEXT I
50 PRINT VET (Ø, VET(2),
   VET (4), VET(6), VET(8)
```

5.2.
```
10 DIM A(10)
20 DIM B(10)
30 FOR I=1 TO 10
40 LET A(I)=I
50 LET B(I)=10-I
60 NEXT I
70 LET SOMA=Ø
```

Tabelas e matrizes

O comando DIM também pode ser usado para definir na memória outro tipo de conjunto: as tabelas ou matrizes.

Uma tabela consiste em um arranjo de elementos de memória dispostos em duas dimensões. Por exemplo, as notas dos alunos de uma classe, nas várias matérias:

	MATERIAS		
	1 MATEMATICA	2 PORTUGUES	3 HISTORIA
ALUNO Nº			
1	8.5	5	9
2	6.5	10	4.5
3	7.5	8	5
4	2	3	1.5

Uma tabela é organizada em *linhas* e *colunas*. No exemplo acima, as linhas são os alunos. Cada linha contém, no exemplo, três colunas, correspondentes às matérias.

Na memória do computador, esses dados seriam armazenados da seguinte forma:

		COLUNAS		
		1	2	3
LINHAS	1	8.5	5	9
	2	6.5	10	4.5
	3	7.5	8	5
	4	2	3	1.5

Um elemento de uma tabela ou matriz é indicado por *dois* subscritos, em vez de um: o número da linha, seguido pelo número da coluna.

Por exemplo:

O ELEMENTO na LINHA 3 e COLUNA 2 é igual a 8.

Ele corresponde à nota de Português do aluno nº 3.

Da mesma forma como nos conjuntos numéricos que você aprendeu até agora (*vetores*) podemos usar um único nome de variável para indicar uma tabela. Por exemplo, o conjunto acima poderia ser chamado de NOTA.

É necessário usarmos a declaração DIM, também no caso das tabelas, para indicar ao computador qual é a área de memória que ele precisa reservar para armazenar o conjunto.

No exemplo acima, usaríamos a declaração.

DIM NOTA (4,3)

DIM NOTA: nome da variável
(4,3): nº de linhas e nº de colunas

A referência a um elemento de um conjunto desse tipo é feita da seguinte forma:

PRINT NOTA (3,2)

PRINT NOTA: nome da variável
(3,2): índice da linha e índice da coluna

Dizemos que o conjunto NOTA acima é um conjunto *bidimensional*, isto é, ele tem *duas dimensões* (linhas e colunas).

O conjunto C(5), por exemplo, é então chamado, por analogia, de *unidimensional* (só tem uma dimensão).
Não confunda o termo *dimensão* com o *tamanho* (ou dimensionamento) de um conjunto expresso pela declaração DIM.

DIM A(25) → o conjunto A tem uma dimensão e está dimensionado para 25 elementos

DIM B(20,35) → o conjunto B tem duas dimensões e está dimensionado para 20 linhas e 35 colunas

Na referência a um elemento de uma tabela ou matriz, os subscritos (índices de linha e coluna) também podem ser variáveis ou expressões:

```
PRINT A(X,5)

INPUT B(I,J)

LET (1,1) = B(X+25,Y*14-2)

etc.
```

É preciso lembrar também que o elemento 0 é automaticamente reservado em uma tabela de duas dimensões, com o comando DIM. Por exemplo:

DIM NOTA (4,3)

gera automaticamente os elementos

NOTA (0)

NOTA (0,1), NOTA (0,2) ....
NOTA (0,3)

NOTA (1,0), NOTA (0,2) ....
NOTA (0,3)

É mais fácil entender a correspondência entre subscritos de um conjunto bidimensional, e sua organização na memória, examinando o esquema abaixo:

		COLUNAS			
		0	1	2	3
LINHAS	0	NOTA (0,0)	NOTA (0,1)	NOTA (0,2)	NOTA (0,3)
	1	NOTA (1,0)	NOTA (1,1)	NOTA (1,2)	NOTA (1,3)
	2	NOTA (2,0)	NOTA (2,1)	NOTA (2,2)	NOTA (2,3)
	3	NOTA (3,0)	NOTA (3,1)	NOTA (3,2)	NOTA (3,3)
	4	NOTA (4,0)	NOTA (4,1)	NOTA (4,2)	NOTA (4,3)

Portanto, o número de elementos de uma tabela é igual a:

(número de linhas + 1) × (número de colunas + 1)

DIM NOTA (4,3) tem: (4+1) × (3+1) = 20 elementos

Nesse caso, você já está percebendo que é desvantajoso deixar de utilizar os elementos com subscrito 0 em uma matriz, pois estaríamos desperdiçando muito espaço. No exemplo acima, oito elementos da matriz ficarão em branco (embora ocupem espaço na memória) se não forem usados. Quanto maior a matriz, maior será o número de elementos 0 desperdiçados.

Programando com matrizes

Tal como acontece com os vetores (conjuntos unidimensionais), usamos as alças tipo FOR...NEXT para realizar programações com conjuntos bidimensionais. Nesta seção aprenderemos alguns truques dessas programações.

Vamos tomar como exemplo, inicialmente, o seguinte problema: como podemos programar a entrada das notas de todos os alunos nas três matérias em uma tabela na memória?

Existem várias maneiras de resolver esse problema. Vamos ver as três técnicas principais:

1. Sem alças

O programa para entrada dos dados poderia ser feito assim:

```
10 DIM NOTA (4,3)                    ----► dimensionamento da tabela
20 PRINT "ESTUDANTE 1, NOTA 1 ?"     ----► mensagem para pedir nota
30 INPUT NOTA (1, 1)                 ----► entrada da nota na tabela
40 PRINT "ESTUDANTE 1, NOTA 2 ?"     ----► mensagem para pedir outra nota
50 INPUT NOTA (1, 2)                 ----► entrada da nota na tabela.
        .
        .
        .
      etc.
```

O programa teria que repetir doze vezes cada linha PRINT e INPUT acima, para as três notas de cada aluno da classe.

Já vimos que esse método é bastante ineficiente, produz programas muito longos e repetitivos, e não pode ser expandido com facildiade. Somente se justifica seu uso quando o número de elementos da matriz a ser entrada é muito pequeno.

2. Com uma alça

Evidentemente, uma maneira mais fácil e sintética de se resolver o problema acima é usar uma alça FOR...NEXT para gerar, repetitivamente, os índices do conjunto NOTA, para cada aluno.

```
10 DIM NOTA (4, 3)
20 FOR I=1 TO 4                      ----► repetir para alunos de 1 a 4
30 PRINT "NOTA 1 DO ALUNO NO.": I    ----► mensagem para pedir nota 1
40 INPUT NOTA (I, 2)                 ----► entra nota 1 do aluno I
50 PRINT "NOTA 2 DO ALUNO NO.";I     ----► mensagem para pedir nota 2
60 INPUT NOTA (I, 3)                 ----► entra nota 2 do aluno I
70 PRINT "NOTA 3 DO ALUNO NO.";I     ----► mensagem para pedir nota 3
80 INPUT NOTA (I, 4)                 ----► entra nota 3 do aluno I
90 NEXT I                            ----► próximo aluno
```

Note que a alça que vai da linha 20 até a linha 90 repete sempre a mesma seqüência de entrada de três notas, para cada um dos quatro alunos.

A variável de controle I da alça é usada para variar o índice de linha da matriz NOTA. A sua ação seria semelhante à geração de doze comandos INPUT sucessivos, assim:

```
I        comando

1     INPUT NOTA (1,1)
      INPUT NOTA (1,2)
      INPUT NOTA (1,3)

2     INPUT NOTA (2,1)
      INPUT NOTA (2,2)
      INPUT NOTA (2,3)

3     INPUT NOTA (3,1)
      INPUT NOTA (3,2)
      INPUT NOTA (3,3)
```

e assim por diante.

Esse método ainda tem também a desvantagem de levar-nos a alterar o programa, se o número de matérias (e de notas por aluno) for maior do que 3. Entretanto, se o número de colunas da matriz for pequeno, a técnica pode ser usada.

3. Com duas alças

Evidentemente, essa é a técnica mais genérica e sintética (ou seja, com ela podemos escrever um programa que faça a mesma coisa que os outros, em um número menor de linhas de BASIC).

Essa técnica utiliza duas alças FOR...NEXT, uma dentro da outra. Uma alça é utilizada para controlar os índices das linhas (estudantes), e a outra, para controlar os índices das colunas (notas):

```
10 DIM NOTA (4, 3)
20 FOR I=1 TO 4                        ---► repetir para alunos de 1 a 4
30 FOR J=1 TO 3                        ---► repetir para notas de 1 a 3
40 PRINT "ALUNO"; I; "NOTA"; J; "?"    ---► mensagem para aluno I, nota J
50 INPUT NOTA (I, J)                   ---► entrada da nota na tabela
60 NEXT J                              ---► próxima nota do aluno I
70 NEXT I                              ---► próximo aluno
```

Veja o que apareceria na tela do computador, quando o programa fosse executado:

```
ALUNO 1 NOTA 1?
ALUNO 1 NOTA 2?
ALUNO 1 NOTA 3?
ALUNO 2 NOTA 1?
ALUNO 2 NOTA 2?
ALUNO 2 NOTA 3?
ALUNO 3 NOTA 1?
      .
      .
      .
     etc.
```

até completar todos os alunos e notas.

A alça externa, com a variável I, serve para aumentar o índice da matriz NOTA, de 1 em 1.

A alça interna, com a variável J, serve para aumentar o índice da matriz NOTA, de 1 em 1.

Desse modo, dizemos que *a alça interna varia (aumenta) mais depressa que a alça externa.*

Em outras palavras, para cada aluno I, o índice J varia de 1 a 3. Portanto, a entrada dos dados é feita *por colunas.*

Poderíamos, neste exemplo, ter adotado uma outra forma de entrada dos dados: entrar as notas de cada um dos alunos, para cada uma das matérias. Por exemplo, primeiro entraríamos todas as notas de Matemática, depis de Português e depois de História.

Esse programa teria uma pequena diferença em relação ao anterior. Qual seria?

```
10 DIM NOTA (4, 3)
20 FOR J=1 TO 3                        ---► repetir para matérias de 1 a 3
30 FOR I=1 TO 4                        ---► repetir para alunos de 1 a 4
40 PRINT "ALUNO"; I; "NOTA"; J; "?"    ---► mensagem para aluno I, nota J
50 INPUT NOTA (I, J)                   ---► entrada da nota na tabela
60 NEXT J                              ---► próxima nota do aluno I
70 NEXT I                              ---► próximo aluno
```

Observe que precisamos mudar apenas as linhas 20 e 30.

Agora a seqüência de mensagens na tela ficaria assim:

```
ALUNO 1 NOTA 1?
ALUNO 2 NOTA 1?
ALUNO 3 NOTA 1?
ALUNO 4 NOTA 1?
ALUNO 1 NOTA 2?
ALUNO 2 NOTA 2?
ALUNO 3 NOTA 2?
ALUNO 4 NOTA 2?
      .
      .
      .
     etc.
```

até completar todos os alunos e notas.

A alça externa, com a variável J, serve para aumentar o índice da matriz NOTA, de 1 em 1.

A alça interna, com a variável I, serve para aumentar o índice da matriz NOTA, de 1 em 1.

Tabelas de qualquer tamanho

Uma grande vantagem do esquema de programação com duas alças é que podemos fazer um programa para qualquer número de alunos ou de matérias, pois o valor limite das alças I e J pode ser entrado como uma variável do programa.

Veja o exemplo abaixo:

```
10 DIM NOTA (50, 15)
12 PRINT "QUANTOS ALUNOS?"
14 INPUT NA                          ---► pergunta número de alunos (NA)
16 PRINT "QUANTAS MATERIAS?"
18 INPUT NM                          ---► pergunta número de matérias
20 FOR I=1 TO NA                     ---► repetir para alunos de 1 a NA
30 FOR J=1 TO NM                     ---► repetir para matérias de 1 a NM
40 PRINT "ALUNO"; I; "NOTA"; J; "?"  ---► mensagem para aluno I, nota J
50 INPUT NOTA (I, J)                 ---► entrada da nota na tabela
60 NEXT J                            ---► próxima nota do aluno I
70 NEXT I                            ---► próximo aluno
```

Observe que demos agora uma capacidade maior à matriz NOTA, de modo a comportar notas de até 50 alunos e 15 matérias.

Novamente, com apenas duas pequenas mudanças nas linhas 20 e 30 do programa, ele continua sendo válido para qualquer número de alunos e de notas (desde que não ultrapassem os limites impostos pela declaração DIM, é claro!)

As alças "embutidas" (ver página 53) podem ser utilizadas também para mostrar o conteúdo dos elementos de uma matriz, por meio do comando PRINT, ou para cálculos (por exemplo, cálculo das médias de cada aluno).

Veja o exemplo abaixo, ainda utilizando a tabela de notas:

```
90 CLS                              ---► limpa a tela
100 FOR I=1 TO 4                    ---► repetir para alunos de 1 a 4
110 FOR J=1 TO 3                    ---► repetir para notas de 1 a 3
120 PRINT "ALUNO"; I; "NOTA"; J;"-"; NOTA (I, J)
140 NEXT J                          ---► próxima nota do aluno I
160 NEXT I                          ---► próximo aluno
```

Este programa mostraria na tela o seguinte resultado (com os dados da tabela de exemplo):

```
ALUNO 1 NOTA 1 = 8.5
ALUNO 1 NOTA 2 = 5
ALUNO 1 NOTA 3 = 9
ALUNO 2 NOTA 1 = 6.5
ALUNO 2 NOTA 2 = 10
ALUNO 2 NOTA 3 = 4.5
ALUNO 3 NOTA 1 = 7.5
        .
        .
        .
       etc.
```

É óbvio que essa não é uma maneira muito elegante de apresentar uma tabela de notas.

Mas, com pequenas modificações no programa, podemos fazer isso facilmente. Observe a diferença:

```
90 CLS                              ---► limpa a tela
100 FOR I=1 TO 4                    ---► repetir para alunos de 1 a 4
105 PRINT "ALUNO"; I; "-";
110 FOR J=1 TO 3                    ---► repetir para notas de 1 a 3
120 PRINT TAB (J*10 +3); NOTA (I,J);
140 NEXT J                          ---► próxima nota do aluno I
150 PRINT
160 NEXT I                          ---► próximo aluno
```

Agora os resultados aparecerão assim

```
ALUNO 1 -    8.5      5     9
ALUNO 2 -    6.5     1Ø     4.5
ALUNO 3 -    7.5      8     5
ALUNO 4 -    2        3     1.5
```

Vamos entender como essa parte do programa funciona:

A linha 1Ø5 escreve na tela a palavra ALUNO, seguida do seu número (no caso, o conteúdo da variável I, que é aumentada de 1 em 1 pela alça da linha 1ØØ). O ponto-e-vírgula colocado no final dessse comando assegura que o próximo comando PRINT escreva na mesma linha (logo após o sinal de subtração).

A linha 12Ø escreve na tela todas as três notas para o aluno I, sempre na mesma linha, pois também tem um ponto-e-vírgula no seu final.

A expressão que usa a função TAB serve para alinhar (colocar nas mesmas colunas, na vertical) cada uma das notas das matérias.

Relembrando, a função TAB *n* vai mover a posição de impressão (cursor) na tela, até a coluna *n*. TAB significa *tabulação*, e funciona exatamente como o tabulador de uma máquina de escrever. É muito útil, portanto, para a programação de *tabelas*.

A expressão entre parênteses será calculada, dentro de cada repetição de I e J, e irá determinar esta nova posição *n*:

J	valor da expressão	(J*1Ø+3)
1	(1 x 1Ø + 3)	13
2	(2 x 1Ø + 3)	23
3	(3 x 1Ø + 3)	33

Portanto, vemos que cada nota começará nas colunas 13, 23 e 33 da mesma linha, respectivamente.

O comando PRINT na coluna 15Ø, Por sua vez, serve para passar o cursor para a linha seguinte. Se ele não fosse colocado aí (ou seja, depis de terminar todas as notas de uma linha, ao final da alça J), as notas do próximo aluno continuariam a ser mostradas na mesma linha do aluno anterior, pois existe um ponto-e-vírgula. Colocando-se o PRINT sozinho, essa linha é encerrada, e passar-se à próxima.

Experimente colocar o programa no computador e retirar a linha 15Ø para ver o que acontece!

Fazendo cálculos em uma tabela

O mesmo esquema de duas alças também pode ser utilizado para realizar cálculos ou comparações lógicas com os elementos de uma matriz.

Vamos imaginar, por exemplo, que queremos saber qual foi a média de todas as notas tiradas para a classe.

Para isso, primeiro vamos ter que somar todas as notas contidas na matriz NOTA, e depois dividir pelo total de notas, certo?

```
19Ø LET SOMA = Ø                      ---→ zerar um acumulador
2ØØ FOR I = 1 TO 4                    ---→ repetir para alunos de 1 a 4
21Ø FOR J = 1 TO 3                    ---→ repetir para notas de 1 a 3
22Ø LET SOMA = SOMA + NOTA (I, J)     ---→ acumular a soma com a nota
23Ø NEXT J                            ---→ próxima nota do aluno I
24Ø NEXT I                            ---→ próximo aluno
25Ø LET MED = SOMA/12                 ---→ calcula a média
26Ø PRINT
27Ø PRINT "MEDIA = "; MED
```

Essa parte do programa, portanto, pegaria cada um dos elementos da matriz NOTA em seguida, através da variação dos índices I e J, e somaria à variável SOMA.

Não precisamos, necessariamente, fazer variar os dois índices da matriz NOTA, através de alças FOR...NEXT.

Vamos imaginar, por exemplo, que queremos saber apenas a médida das notas de todos alunos para uma determinada matéria. Para isto, a variável SOMA deveria ser acumulada apenas com os valores da coluna desejada:

```
290 LET SOMA = Ø                        ---→ zerar um acumulador
300 FOR I = 1 TO 4                      ---→ repetir para alunos de 1 a 4
310 LET SOMA = SOMA + NOTA (I,1)        ---→ acumular a soma com a nota
320 NEXT I                              ---→ próximo aluno
330 LET MED = SOMA/4                    ---→ calcula a média
340 PRINT
350 PRINT "MEDIA DE MATEMATICA = "; MED
```

Note que a linha 31Ø soma apenas os elementos

NOTA (1,1) NOTA (2,1) NOTA (3,1) e NOTA (4,1)

Isso ocorre porque o índice da coluna (no caso, coluna 1) está *fixo* dentro do programa, ou seja, é uma constante que não varia, ao passo que o índice de linha (alunos) varia de 1 a 4, através da alça I da linha 3ØØ.

Para tornar mais geral esta parte do programa, poderíamos perguntar ao usuário de que matéria ele deseja saber a média.

```
280 PRINT "ENTRE QUE MATERIA (1, 2 OU 3);"
285 INPUT M
290 LET SOMA = Ø                        ---→ zerar um acumulador
300 FOR I = 1 TO 4                      ---→ repetir para alunos de 1 a 4
310 LET SOMA = SOMA + NOTA (I,M)        ---→ acumular a soma com a nota
320 NEXT I                              ---→ próximo aluno
330 LET MED = SOMA/4                    ---→ calcula a média
340 PRINT
350 PRINT "MEDIA DE MATERIA "; M;" = "; MED
```

Ou ainda, poderíamos obter automaticamente as médias para todas as matérias, usando uma alça externa para fazer variar os índices de coluna:

```
280 FOR M = 1 TO 3                      ---→ repetir para matérias de 1 a 3
290 LET SOMA = Ø                        ---→ zerar um acumulador
300 FOR I = 1 TO 4                      ---→ repetir para alunos de 1 a 4
310 LET SOMA = SOMA + NOTA (I,M)        ---→ acumular a soma com a nota
320 NEXT I                              ---→ próximo aluno
330 LET MED = SOMA/4                    ---→ calcula a média
340 PRINT
350 PRINT "MEDIA DE MATERIA "; M;" = "; MED
360 NEXT M                              ---→ próxima matéria
```

Observe que estamos usando as mesmas variáveis, SOMA e MED, para o cálculo da média de cada nota. Isso realmente não tem importância, pois não queremos armazenar os valores das médias de todas as matérias. A variável SOMA, portanto, deve ser zerada *dentro* da alça externa, e *fora* da alça interna.

m programa completo

Vamos fazer agora um programa completo, utilizando o exemplo das notas de uma classe.

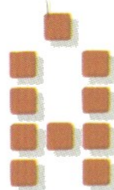
Neste programa queremos fazer o seguinte:

1. entrar as notas de cada aluno, em todas as matérias;

2. mostrar essas notas na tela, em forma de tabela. Aproveitamos para calcular e mostrar a médida aritmética de cada aluno, nessa tabela;

3. calcular e mostrar a médida de cada matéria.

Vamos lá:

```
10 REM = =============
20 REM = BOLETIM DE NOTAS DA
         CLASSE
30 REM = BASIC PARA TK 85
40 REM = R. SABBATINI ABRIL 1985
50 REM = ============
60 REM
70 REM **** INICIALIZACAO ****
80 REM
90 DIM NOTA(50, 15)
100 CLS
110 PRINT "CALCULO DO
    BOLETIM DE NOTAS"
120 PRINT "=========================="
130 REM
140 REM **** ENTRADA DE
    PARAMETROS ****
150 REM
160 PRINT
170 PRINT "NUMERO DE ALUNOS
    NA CLASSE ?";
180 INPUT NA
190 IF NA > 50 THEN GOTO 170
200 PRINT NA
210 PRINT "NUMERO DE
    MATERIAS ?";
220 INPUT NM
230 IF NM > 15 THEN GOTO 210
240 PRINT NM
250 REM
260 REM **** ENTRADA DE
    DADOS ****
270 REM
280 PRINT "ENTRE AS NOTAS DOS
    ALUNOS:"
290 PRINT
300 FOR I=1 TO NA
310 PRINT "ALUNO Nº ";I
320 FOR J=1 TO NM
330 PRINT TAB (7);"NOTA DA
    MATERIA ";J;" ?";
340 INPUT NOTA(I,J)
350 IF NOTA(I,J) <0 THEN
    GOTO 340
360 IF NOTA (I, J) > 10 THEN
    GOTO 340
370 PRINT NOTA(I,J)
380 NEXT J
390 REM
400 REM **** MOSTRA NOTAS NA
    TELA ***
410 REM **** E CALCULA
    MEDIAS ****
420 REM
430 CLS
440 FOR I=1 TO NA
450 PRINT "ALUNO ";I;" - ";
460 LET SOMA=0
470 FOR J=1 TO NM
480 PRINT TAB(10+5*J);NOTA(I,J);
490 LET SOMA=SOMA+NOTA(I,J)
500 NEXT J
510 PRINT " MED ";SOMA/NM
520 NEXT I
530 REM
540 REM **** CALCULA MEDIA
    DAS MATERIAS ****
550 REM
560 FOR J=1 TO NM
570 LET SOMA=0
580 FOR I=1 TO NA
590 LET SOMA=SOMA+NOTA(I,J)
600 NEXT I
610 PRINT
620 LET MED=SOMA/NA
630 PRINT "MEDIA NA MATERIA
    ";J;" = ";MED
640 NEXT I
650 STOP
```

nálise do programa

Vamos analisar o programa linha por linha, agora, para podermos entender o que elas fazer:

- **10 a 50** - Nestas linhas colocamos um cabeçalho interno de identificação do programa (autor, data, computador etc.).
- **90** - Dimensionamento do conjunto NOTA.
- **100 a 120** - Mostrar título do programa na tela (para que serve).
- **170 a 180** - Pergunta quantos alunos tem, e entra na variável NA.
- **190** - Testa se NA é maior do que o permitido pelo comando DIM. Se isto acontecer, pergunta novamente.
- **200 a 210** - Pergunta quantas matérias tem, e entra na variável NM.
- **220** - Testa se NM é maior do que o permitido pelo comando DIM. Se isso acontecer, pergunta novamente.
- **280 a 290** - Dá instruções para entrada de dados.
- **300** - Alça de repetição dos alunos.
- **310** - Mostra identificação do aluno a ser entrado.
- **320** - Alça de repetição das matérias por aluno.
- **330 a 340** - Pergunta nota.
- **350 a 360** - Testa se a nota está entre 0 e 10. Em caso contrário, pergunta a nota de novo.
- **440 a 520** - Parte do programa que mostra as notas na tela. Na linha 460, zera o somador de notas de um aluno. Na linha 490, efetua esta soma, e na linha 510 mostra a média.
- **560 a 640** - Parte do programa que calcula as médias das matérias.

Coloque agora o programa no computador e teste com os dados da tabela de exemplo deste capítulo.

EXERCÍCIOS

1. O conjunto DADOS, com cinco linhas e quatro colunas, contém os seguintes elementos:

		COLUNAS			
		1	2	3	4
LINHAS	1	12	23.4	16.7	100
	2	5	6.7	9.2	12
	3	14	6.1	1.1	25
	4	9	45.8	10.9	13
	5	1	0	32	43

Responda às seguintes questões (exemplo de resposta na primeira linha):

1.1. Quais são os subscritos de linha e coluna dos elementos cujo conteúdo é:

23.4 - Resposta: DADO (1,2)

14 - " ______________

45.8 - " ______________

32 - " ______________

43 - " ______________

100 - " ______________

1.2. Escreva uma declaração DIM para o conjunto DADOS acima.

Resposta: ______________

1.3. Indique o valor contido nos elementos abaixo:

DADO(2,1): ______________
DADO(3,3): ______________
DADO(DADO(2,1),DADO(4,1)): _____
DADO(4,DADO(4,1)): ______________
DADO(5,5): ______________
DADO(1,5): ______________
DADO(5,1): ______________
DADO(4,1): ______________

2. Escreva um programa com a alça FOR NEXT, para preencher com dados os elementos do conjunto DADO (por meio de um comando INPUT).

3. Escreva um programa para preencher com zeros todos os elementos do conjunto DADO.

4. Explique o que realizam os programas abaixo, e qual o resultado mostrado pelo comando PRINT na linha 100, tomando como dados o conjunto DADOS acima:

4.1.
```
10 PR=1
15 LET C=3
20 FOR L=1 TO 5
30 LET PROD=
   PROD*DADO(L,C)
40 NEXT L
100 PRINT PROD
```

4.2.
```
10 LET S=0
15 FOR I=2 TO 4 STEP 2
20 FOR J=I TO 4
30 LET S=S+DADO(I,J)
40 NEXT J
50 NEXT I
60 PRINT S
```

5. Faça um programa para copiar uma das linhas da matriz DADO, para um vetor (conjunto unidimensional) chamado COPIA.

6. Faça um programa para copiar uma das colunas da matriz DADO, para um vetor chamado COPIA.

7. Escreva um programa para preencher com o valor 1 todos os elementos da diagonal principal de um conjunto A(10,10).

Dica: Diagonal principal é o conjunto de todos os elementos da matriz que têm o mesmo índice de linha e de coluna. Por exemplo: os elementos A(1,1), A(2,2), A(3,3)... etc. estão na diagonal principal.

8. Indique qual o erro nos programas abaixo:

8.1.
```
10 DIM A(7,6)
20 FOR I=0 TO 7
30 LET A(I,7)=1
40 NEXT I
```

8.2.
```
10 FOR J=2 TO 100 STEP 3
15 LET B(J,10)=B(J,1)*2
20 NEXT J
30 DIM B(100,10)
```

8.3.
```
10 DIM X(50,10)
20 FOR I=1 TO 10
30 FOR J=1 TO 50
40 LET X(I,J)=0
50 NEXT I
60 NEXT J
```

9. Indique se as linhas de programa abaixo estão certas (C) ou erradas (E). Caso estiverem erradas, explique onde está o erro:

```
PRINT XX(A+5,10)

LETC(1;2)=896

PRINT A218(B5,B5+3 - T*4)

PRINT CONJ(4,5)

PRINT B(67000,2)

PRINT X(3.2,4.55)
```

10. Modifique o programa completo de cálculo de boletim de notas, da seção anterior, de modo a realizar as seguintes funções adicionais:

a. Calcular a média global da classe (todas as notas).

b. Indicar, ao lado da média de cada aluno, se ela está acima ou abaixo da média global da classe.

Listas de nomes

Também podemos programar em BASIC uma variável conjunta alfanumérica. No exemplo dos times, poderíamos então armazenar os nomes dos times em uma lista, assim:

```
25 LET TIME$(1)="PALMEIRAS"
26 LET TIME$(2)="SANTOS"
27 LET TIME$(3)="CORINTHIANS"
28 LET TIME$(4)="FLAMENGO"
29 LET TIME$(5)="FLUMINENSE"
```

e assim por diante.

Dimensionar um conjunto literal como este é um pouco mais complicado do que dimensionar um conjunto numérico (um vetor).

Como você deve se lembrar, um *cordão alfanumérico ("string")* é armazenado no computador como uma série de memórias interligadas. Ou seja, uma variável literal simples *já é um conjunto* de memórias.

Normalmente, podemos encher uma variável literal simples com até 255 caracteres, sem necessidade de usar o comando DIM (ou seja, é diferente do conjunto numérico, que só permite dez elementos nessas condições).

Se usarmos o comando DIM com uma variável literal simples, *estaremos automaticamente limitando o número de caracteres que esta variável irá aceitar*. Por exemplo:

```
1Ø DIM LETRAS$(1Ø)
2Ø LET LETRAS$="ABCDEFGHIJKL
        MNOPQRSTUVXZ"
```

Neste caso, o comando

```
3Ø PRINT LETRAS$
```

irá mostrar apenas este resultado (as dez primeiras letras foram aceitas no comando LET da linha 2Ø):

ABCDEFGHIJ

Para dimensionar um *conjunto literal*, portanto, podemos utilizar uma tabela como se fosse uma página com várias linhas de tamanho fixo:

```
1Ø DIM T$(5,15)
```

Isso equivale a um *conjunto de duas dimensões*, organizado assim:

ÍNDICE   ESPAÇO DIMENSIONADO

Ø
1
2
3
4
5

O primeiro número entre parênteses no comando DIM especifica o *número de linhas* que constam do conjunto, ou seja, quantos *nomes separados* eu poderei colocar na minha lista.

O segundo número entre parênteses (depois da vírgula) corresponde ao *número de colunas* que constam do conjunto, ou seja, quantos *caracteres*, no máximo, poderá ter cada nome da lista.

Como no caso dos conjuntos numéricos, existe um elemento Ø que é reservado automaticamente (você pode usá-lo ou não, à vontade).

A única diferença em relação a um conjunto numérico é que não existe a coluna numerada zero, ou seja, ela não é ocupada por um caractere do cordão.

Preenchendo com os nomes dos times do exemplo acima, teríamos:

0															
1	P	A	L	M	E	I	R	A	S						
2	S	A	N	T	O	S									
3	C	O	R	I	N	T	H	I	A	N	S				
4	F	L	A	M	E	N	G	O							
5	F	L	U	M	I	N	E	N	S	E					

Você deve ter notado dois fatos interessantes aqui:

1. A linha do começo do conjunto (linha Ø) está vazia.

2. Sobram espaços em branco depois dos nomes dos times, em cada linha, se estes tiverem menos do que os quinze caracteres dimensionados.

O primeiro fato pode ser explicado porque tínhamos colocado os comandos:

```
25 LET TIME$(1)="PALMEIRAS"
26 LET TIME$(2)="SANTOS"
27 LET TIME$(3)="CORINTHIANS"
28 LET TIME$(4)="FLAMENGO"
29 LET TIME$(5)="FLUMINENSE"
```

Mas nada impediria que colocássemos:

```
24 LET TIME$(Ø)="SAO PAULO"
```

E, neste caso, a linha Ø também seria ocupada:

0	S	A	O		P	A	U	L	O						
1	P	A	L	M	E	I	R	A	S						
2	S	A	N	T	O	S									
3	C	O	R	I	N	T	H	I	A	N	S				
4	F	L	A	M	E	N	G	O							
5	F	L	U	M	I	N	E	N	S	E					

Dando nome aos bois

Vamos refazer agora o programa de entrada dos pontos ganhos por time de futebol, colocando também os nomes dos times.

Um programa para entrar os nomes dos times, por exemplo, seria assim:

```
10 DIM TIME$(5,15)
20 FOR I=1 TO 5
30 PRINT "NOME DO TIME Nº";I;" ?";
40 INPUT TIME$(I)
50 NEXT I
```

Note que o *número de subscritos na declaração DIM* é diferente do *número de subscritos na declaração INPUT*, quando se trata de conjuntos de cordões. Isso não acontece em conjuntos numéricos: ambos os números precisam ser iguais.

Podemos aproveitar a alça de entrada do programa dos times de futebol para colocar também duas linhas que possibilitem a entrada do nome:

```
10 PRINT "CAMPEONATO
   BRASILEIRO DE FUTEBOL"
20 "TABELA DE PONTOS GANHOS"
30 PRINT
35 DIM TIME$(5,15)
40 REM
50 REM ::: ENTRADA DE DADOS
70 REM
70 PRINT "ENTRE O NUMERO DE
   PONTOS GANHOS"
75 PRINT "PARA CADA TIME DA
   TABELA:"
80 PRINT
85 FOR N=1 TO 5
90 PRINT "NOME DO TIME
   NUMERO ";N;" ?";
95 INPUT TIME$(N)
96 PRINT TIME$(N)
98 PRINT "NUMERO DE PONTOS
   GANHOS ?";
100 INPUT T(N)
110 PRINT T(N)
120 NEXT N
130 REM
140 REM ::: CALCULO DA MEDIA
150 REM
160 LET SOMA=Ø
170 FOR N=1 TO 5
180 LET SOMA=SOMA + T(N)
190 NEXT N
200 REM
```

```
210 REM ::: MOSTRA RESULTADOS
220 REM
400 LET MEDIA = SOMA/5
410 PRINT
420 PRINT "MEDIA DOS PONTOS
    GANHOS = ";MEDIA
430 STOP
```

Eis aqui um exemplo de utilização do programa:

```
CAMPEONATO BRASILEIRO DE
FUTEBOL
TABELA DE PONTOS GANHOS

ENTRE O NUMERO DE PONTOS
GANHOS PARA CADA TIME
DA TABELA:

NOME DO TIME NUMERO 1 ?
PALMEIRAS
NUMERO DE PONTOS GANHOS ? 23
NOME DO TIME NUMERO 2 ?
SANTOS
NUMERO DE PONTOS GANHOS ? 12
NOME DO TIME NUMERO 3 ?
CORINTHIANS
NUMERO DE PONTOS GANHOS ? 9
NOME DO TIME NUMERO 4 ?
FLAMENGO
NUMERO DE PONTOS GANHOS ? 7
NOME DO TIME NUMERO 5 ?
FLUMINENSE
NUMERO DE PONTOS GANHOS ? 4

MEDIA DOS PONTOS GANHOS = 11
```

Você deve estar perguntando a esta altura: Por que entrar os nomes dos times, se eles não são usados em nenhum outro lugar do programa?

Realmente, se for só para informar no momento de entrada de dados, é desnecessário entrar os nomes dos programas.

Porém, uma vez que já temos os nomes armazenados no conjunto TIME$, podemos usá-los para outros fins.

Vamos supor que queremos modificar o programa acima, de modo a mostrar na tela o seguinte resultado:

TIME	PONTOS	DIFERENÇA MEDIA
1. PALMEIRAS	23	12
2. SANTOS	12	1
3. CORINTHIANS	9	-2
4. FLAMENGO	7	-4
5. FLUMINENSE	4	-7

Para fazer isso, tiramos a linha 430 do programa anterior e adicionamos:

```
430 CLS
440 PRINT TAB( 5);"TIME";
    TAB(15);"PONTOS";
450 PRINT TAB(22);"DIFERENCA
    MEDIA"
460 FOR I=1 TO 5
470 PRINT I;". ";TIME$(I);
    TAB(7);T(I);TAB(17);MEDIA - T(I)
480 NEXT I
490 STOP
```

A expressão da linha 470 mostra os dados de entrada e a diferença dos pontos ganhos por cada time, em relação à média calculada.

Portanto, para realizar este tipo de cálculo, bem como mostrar os resultados, é necessário usar conjuntos para armazenamento intermediário dos dados.

Listas internas de nomes

Muitos programas utilizam listas de nomes e outros dados, que podem ser definidos como fixos, dentro dos mesmos, em vez de serem entrados por meio de comandos INPUT.

Em muitos computadores, existem dois comandos (READ e DATA) que facilitam bastante a definição interna de tais listas. Entretanto, os computadores com a lógica SINCLAIR (por exemplo, o TK83, TK85, CP200 etc., no Brasil) não têm estes comandos no seu interpretador BASIC. Por isso, temos que usar alguns truques e assim contornar essa falta.

O mais comum, se o número de elementos dos conjuntos não for muito grande, é defini-los no programa por meio de declarações LET.

Vamos tomar um exemplo.

Suponhamos que o dono de uma loja de frios queira emitir uma nota de despesas para cada cliente que efetua compras na sua loja. Os produtos que ele tem para vender são fixos, e podem constar do programa. Para cada cliente ele digita a quantidade comprada e o preço por quilograma da mercadoria. Em seguida, aparece a nota na tela, com o total das despesas, bem como o nome e valor de cada mercadoria comprada.

Vamos definir o conjunto M$, que contém os nomes das mercadorias, e os conjuntos Q e PR, que contêm quantidade e preço, respectivamente.

```
10 CLS
20 PRINT "EMISSAO DE NOTA DE
   DESPESAS"
30 PRINT "CASA DE FRIOS"
   "AZEITONA DOURADA" LTDA"
40 PRINT
50 DIM M$(5,20)
55 DIM TT(5)
60 LET M$(0)="AZEITONAS
   PRETAS"
70 LET M$(1)="AZEITONAS
   VERDES"
80 LET M$(2)="PRESUNTO
   GORDO"
90 LET M$(3)="PRESUNTO
   MAGRO"
100 LET M$(4)="MOZARELA"
110 LET M$(5)="QUEIJO PRATO"
120 REM
130 REM ***** ENTRADA DOS
    DADOS DA COMPRA *****
140 REM
150 PRINT "DIGITE A QUANTIDADE
    COMPRADA E DEPOIS"
160 PRINT "O PREÇO POR
    QUILOGRAMA PARA
   CADA MER - "
170 PRINT "CADORIA APRESENTA-
    DA NA TELA. DIGITE 0"
180 PRINT "SE O CLIENTE NAO
    COMPROU A MERCADORIA"
190 PRINT
200 FOR I=1 TO 5
210 PRINT M$(I); " ? "
220 INPUT Q(I)
230 IF Q(I)=0 THEN GOTO 250
240 INPUT PR(I)
250 NEXT I
260 REM
270 REM ***** IMPRESSAO DA
    NOTA *****
```

```
280 REM
290 CLS
300 PRINT "                    NOTA
    DE DESPESAS"
310 PRINT
320 PRINT "MERCADORIA";
    TAB(17);"QUANT PRECO"
325 LET TOT=0
330 FOR I=1 TO 5
340 IF Q(I)=0 THEN GOTO 360
350 PRINT M$(I);TAB(17);Q(I);
    TAB(22);Q(I)*PR(I)
355 LET TOT=TOT+Q(I)*PR(I)
360 NEXT I
365 PRINT "TOTAL";TAB(22);TOT
370 PAUSE 600
380 CLS
390 GOTO 150
```

O comando IF da linha 230 serve para testar se a quantidade comprada foi zero. Neste caso, o programa não pede o preço depois.

O comando IF da linha 350 testa também se a mercadoria tem valor zero de quantidade, e passa para a linha 350 de apresentação, neste caso.

EXERCÍCIOS

1. Modifique o programa de emissão de nota de despesas, de modo a ter as seguintes funções adicionais:

a. entrar a data de emissão da nota e o nome do cliente, para mostrar na tela de emissão;

b. armazenar o valor total de cada nota, cumulativamente em uma variável RENDA, e mostrar este valor se o nome do cliente for "FIM".

2. Modifique o programa de cálculo de boletim de notas (página 87), a fim de poder entrar e mostrar o nome de cada aluno.

Listas mais complexas

Muitos programas necessitam de várias listas alfanuméricas relacionadas entre si.

Vamos tomar como exemplo o seguinte problema de programação:

Queremos fazer um programa para entrar pelo teclado e armazenar na memória do computador uma caderneta de endereços de amigos e conhecidos. Gostaríamos que neste programa os dados relativos a cada pessoa fossem armazenadas em listas separadas:

NOME$ - lista dos nomes das pessoas

ENDEREÇO$ - lista dos endereços

TELEFONE$ - listas dos telefones

Assim, por exemplo, os dados sobre as pessoas seriam armazenadas nos elementos das listas:

INDICE	NOME$	ENDEREÇO$	TELEFONE$
1	Maria Alice	Rua do Bonfim, 122	234-3443
2	Roberto	Av. da Praia, 12/456	123-5566
3	Susana	Pça. dos Coqueiros, 9	32-4567
4	Tio Zé	Rua do Bonde, 334	64-5980
		. . . etc.	

Note que os elementos NOME$ (5), ENDEREÇO$ (5) e TELEFONE$ (5), por exemplo, contêm dados da pessoa nº 5 em nossa caderneta.

Organizando os dados dessa forma, será mais fácil programar as diversas funções que caracterizam um programa de *cadastramento de informações,* como é o caso da nossa caderneta de endereços.

Programas desse tipo têm normalmente as seguintes funções (partes do programa que realizam daterminadas tarefas em relação aos conjuntos de dados):

- incluir na caderneta os dados de uma pessoa;
- procurar o telefone ou endereço de uma determinada pessoa;
- modificar os dados de uma pessoa (por exemplo, se ela mudou de telefone ou de endereço).

Uma forma bastante simples e muito utilizada de programar a opção do usuário por estas funções é um menu.

Na gíria de computação, um menu é uma lista das opções disponíveis de processamento dentro de um determinado programa, apresentada ao usuário para que ele selecione uma.

Em nosso programa, um exemplo prático de menu seria:

```
       AGENDA TK                  <---------- título do menu
 CADERNETA DE ENDERECOS

      COMANDOS:
=================
1   INCLUIR UM NOME               <---------- lista de comandos
2   MODIFICAR DADOS
3   INFORMAR DADOS
4   FIM                           <---------- comando para terminar o
=================                             programa

DIGITE UM NUMERO:                 <---------- mensagem de solicitação
```

A apresentação deste menu geralmente é colocada logo no início do programa:

```
40 CLS
50 PRINT "              AGENDA TK"
60 PRINT "CADERNETA DE
   ENDERECOS"
70 PRINT
80 PRINT "           COMANDOS:"
90 PRINT "===============
   ==========="
100 PRINT "1            INCLUIR
    UM NOME"
120 PRINT "2          MODIFICAR
    DADOS"
130 PRINT "3           INFORMAR
    DADOS"
150 PRINT "4               FIM"
160 PRINT "===============
    ==========="
170 PRINT
180 PRINT "DIGITE UM
    NUMERO:"
190 IMPUT OPCAO
200 IF OPCAO < 1 THEN
    GOTO 190
210 IF OPCAO > 4 THEN
    GOTO 190
```

A linha 190 fica aguardando até que o usuário digite um número correspondente à opção que ele deseja executar.

As linhas 200 e 210 testam se o número digitado está situado entre 1 e 4 (as opções disponíveis). Se for digitado um número diferente destes, o programa retornará à linha 190, até que a entrada esteja correta.

A partir daí, a estrutura do programa poderá ser a seguinte:

Explicando:

Após a escolha do usuário, o programa desvia para o segmento apropriado, que irá realizar aquela função. Depois de terminá-la, retorna ao início do programa, para apresentar de novo o menu (exceto quando o usuário escolhe terminar o programa).

Essa estrutura permite que o usuário do programa realize facilmente uma série de tarefas, na ordem que julgar mais interessante, sem necessidade de "decorar" comandos por extenso ou ser obrigado a seguir uma ordem inflexível de processamento.

Continuando a programação, devemos dimensionar previamente os conjuntos alfanuméricos que serão utilizados:

```
37 DIM NOME$ (50,25)
38 DIM ENDERECO$ (50,25)
39 DIM TELEFONE$ (50,10)
```

Note que dimensionamos a caderneta para um máximo de 50 nomes de 25 caracteres cada, mais os endereços, também de 25 caracteres cada, e os telefones, com 10 caracteres cada.

Portanto, a quantidade de memória a ser ocupada por esses conjuntos será de:

$50 \times 25 + 50 \times 25 + 50 \times 10 =$ 3 000 caracteres

Portanto, *atenção*: o programa não poderá ser utilizado em microcomputadores com dois kbytes de memória (versão básica do TK 82 e do TK 83); apenas em computadores de maior capacidade.

Para armazenar um número maior ou menor de endereços, basta alterar os valores nos comandos DIM.

Observe também que cada informação tem um limite máximo em número de caracteres, e estes serão "podados" automaticamente se ultrapassaram esse limite de comprimento.

Agora, podemos verificar qual foi a opção dada pelo usuário e desviar o programa de acordo com o resultado:

```
250 IF OPCAO = 1 THEN GOTO
    300
260 IF OPCAO = 2 THEN GOTO
    400
270 IF OPCAO = 3 THEN GOTO
    500
280 IF OPCAO = 4 THEN STOP
```

Agora, a partir da linha 300, colocamos o setor de entrada de dados. Vamos definir a variável NUM para contar o número de endereços armazenados na caderneta. Inicialmente, é claro, ele será zero. A cada entrada na caderneta de uma nova pessoa, esse número será incrementado por mais um:

```
300 LET NUM = NUM + 1
310 IF NUM > 25 THEN GOTO 900
320 CLS
330 PRINT "ENDERECO NO.";NUM
335 PRINT "==============="
340 PRINT "NOME:"
345 INPUT NOME$ (NUM)
350 PRINT "ENDERECO:"
355 INPUT ENDERECO$ (NUM)
360 PRINT "TELEFONE:"
365 INPUT TELEFONE$ (NUM)
370 GOTO 40
```

A linha 31Ø tem por objetivo impedir a entrada de mais de 25 nomes, pois assim "estouraria" a capacidade dimensionada para os conjuntos.

A seguir programamos o segmento de modificação de dados. Inicialmente mostramos no alto da tela como estão os dados da pessoa solicitada, e depois pedimos os dados novos:

```
400 CLS
410 PRINT "ENTRE O NUMERO DO
     REGISTRO A SER
     MODIFICADO:"
415 INPUT X
420 IF X<1 THEN GOTO 415
425 IF X>25 THEN GOTO 415
430 PRINT
435 PRINT "ENDERECO NO.";X
440 PRINT "================"
445 PRINT "NOME:";NOME$(X)
450 PRINT "ENDERECO:";
     ENDERECO$(X)
455 PRINT "TELEFONE:";
     TELEFONE$(X)
460 PRINT "NOVO NOME:"
465 INPUT NOME$(X)
470 PRINT "NOVO ENDERECO:"
475 INPUT ENDERECO$(X)
480 PRINT "NOVO TELEFONE:"
485 INPUT TELEFONE$(X)
490 GOTO 40
```

Finalmente, podemos programar o segmento de procura. Nesse caso, o usuário deve fornecer o nome da pessoa desejada (exatamente como está gravado no conjunto NOME$), e o programa procurará:

```
500 CLS
510 PRINT "DIGITE O NOME DA
     PESSOA:";
520 INPUT X$
530 PRINT X$
540 FOR I=1 TO NUM
550 IF X$<>NOME$(I) THEN
     GOTO 580
560 PRINT "ENDERECO:";
     ENDERECO$(I)
565 PRINT "TELEFONE:";
     TELEFONE$(I)
570 PAUSE 600
575 GOTO 40
580 NEXT I
590 PRINT "NAO CONHECO ESTE
     NOME"
595 GOTO 510
```

Se todos os nomes forem testados (linha 55Ø) e não for achado o pedido, o programa avisa ao usuário, após terminar a alça da linha 58Ø.

Temos agora o nosso programa completo, com os comentários nos lugares apropriados:

```
10 REM ********************************
15 REM * PROGRAMA AGENDA 1.00         *
20 REM * PARA LOGICA SINCLAIR         *
25 REM * PROGRAMA R. SABBATINI        *
30 REM ********************************
32 REM
34 LET NUM=0
37 DIM NOME$(50,25)
38 DIM ENDERECO$(50,25)
39 DIM TELEFONE$(50,10)
40 CLS
50 PRINT "            AGENDA TK"
60 PRINT " CADERNETA DE ENDERECOS"
70 PRINT
80 PRINT "            COMANDOS:"
90 PRINT "=========================="
100 PRINT "1          INCLUIR UM NOME"
120 PRINT "2          MODIFICAR DADOS"
130 PRINT "3          INFORMAR DADOS"
150 PRINT "4          FIM"
160 PRINT "=========================="
170 PRINT
172 REM
174 REM ***** SOLICITAR E TESTAR OPCAO *****
176 REM
180 PRINT "DIGITE UM NUMERO:"
190 INPUT OPCAO
200 IF OPCAO<1 THEN GOTO 190
210 IF OPCAO>4 THEN GOTO 190
212 REM
214 REM ***** DIMENSIONAR CONJUNTOS *****
216 REM
242 REM
244 REM ***** DESVIAR PARA OPCAO *****
246 REM
250 IF OPCAO=1 THEN GOTO 300
270 IF OPCAO=2 THEN GOTO 400
280 IF OPCAO=3 THEN GOTO 500
290 IF OPCAO=4 THEN STOP
292 REM
294 REM ***** ADICIONAR NOMES *****
296 REM
300 LET NUM=NUM+1
310 IF NUM>25 THEN GOTO 900
320 CLS
330 PRINT "ENDERECO NO.";NUM
335 PRINT "================"
340 PRINT "NOME:"
345 INPUT NOME$(NUM)
350 PRINT "ENDERECO:"
355 INPUT ENDERECO$(NUM)
360 PRINT "TELEFONE:"
365 INPUT TELEFONE$(NUM)
370 GOTO 40
380 REM
382 REM ***** MODIFICAR DADOS *****
384 REM
400 CLS
410 PRINT "ENTRE O NUMERO DO REGISTRO
     A SER MODIFICADO:"
415 INPUT X
420 IF X<1 THEN GOTO 415
425 IF X>25 THEN GOTO 415
430 PRINT
435 PRINT "ENDERECO NO.";X
440 PRINT "=============="
445 PRINT "NOME:";NOME$(X)
```

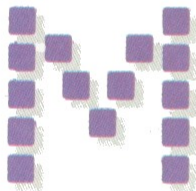
```
450 PRINT "ENDERECO:"; ENDERECO$(X)
455 PRINT "TELEFONE:"; TELEFONE$(X)
460 PRINT "NOVO NOME:"
465 INPUT NOME$(X)
470 PRINT "NOVO ENDERECO:"
475 INPUT ENDERECO$(X)
480 PRINT "NOVO TELEFONE:"
485 INPUT TELEFONE$(X)
490 GOTO 40
492 REM
494 REM ***** PROCURA DADOS *****
496 REM
500 CLS
510 PRINT "DIGITE O NOME DA PESSOA:";
520 INPUT X$
530 PRINT X$
540 FOR I = 1 TO NUM
550 IF X$ < > NOME$(I) THEN GOTO 580
555 FOR J = 2 TO 3
560 PRINT "ENDERECO:"; ENDERECO$(I)
565 PRINT "TELEFONE:"; TELEFONE$(I)
570 PAUSE 600
575 GOTO 40
580 NEXT I
590 PRINT "NAO CONHECO ESTE NOME"
595 GOTO 510
```

Digite este programa no computador e teste-o com vários dados.

EXERCÍCIOS

Modifique o programa acima para incluir as seguintes funções:

1 - exclusão de um nome da caderneta (*dica*: o programa solicita o número do registro a ser excluído e em seguida coloca espaços em branco nos conjuntos correspondentes);

2 - listagem de todos os nomes colocados na caderneta (*dica*: coloque um registro em cada linha da tela, começando com o número do registro, depois o nome, o endereço e o telefone).

atrizes alfanuméricas

Da mesma forma como podemos ter conjuntos numéricos bidimensionais em BASIC, também é possível armazenar dados literais em conjuntos desse tipo.

Essa possibilidade torna-se bastante útil quando desejamos programar listas que têm várias informações relacionadas, como no exemplo da seção anterior (caderneta de endereços).

Como você verá, a programação se torna mais fácil, compacta e genérica se usarmos uma única matriz de dados para representar os diferentes *campos de informação* do nosso cadastro.

Como você se lembra, a caderneta tinha os seguintes conjuntos para armazenamento de dados literais:

NOME$ -lista dos nomes das pessoas

ENDEREÇO$ -lista dos endereços correspondentes

TELEFONE$ -lista dos telefones correspondentes

Nesse novo esquema, teríamos uma matriz (chamada DADOS$, por exemplo, com 25 linhas, ou seja, uma para cada registro e três colunas, uma para cada campo).

Assim, por exemplo, os dados sobre as pessoas seriam armazenados nos elementos da matriz:

	COLUNA		
LINHA	1	2	3
1	Maria Alice	Rua do Bonfim, 122	234-3443
2	Roberto	Av. da Praia, 12/456	123-5566
3	Susana	Pça. dos Coqueiros, 9	32-4567
4	Tio Zé	Rua do Bonde, 334	64-598Ø
	.		
	.		
	etc.		

Organizando os dados dessa forma, será ainda mais fácil programar as diversas funções de *cadastramento de informações* da nossa caderneta de endereços.

Na programação, basta que dimensionemos a matriz:

```
220 DIM DADOS$(50,3,25)
```

Observe que dimensionamos a caderneta para um máximo de 50 pessoas (registros), com três campos cada, de 25 registros. O terceiro número da declaração DIM serve para especificar o comprimento de cada dado literal nesta matriz, que, nesse caso, deve ser fixo para todos os campos.

Portanto, a quantidade de memória a ser ocupada por esses conjuntos será de:

$50 \times 25 + 50 \times 25 + 50 \times 25 =$ 3750 caracteres

Para armazenar um número maior ou menor de endereços, basta alterar os valores nos comandos DIM.

Podemos, em seguida, definir os nomes dos campos, utilizando outro conjunto, chamado CAMPOS$.

```
36 DIM CAMPOS$(3)
37 LET CAMPOS$(1) = "NOME"
38 LET CAMPOS$(2) =
   "ENDERECO"
39 LET CAMPOS$(3) =
   "TELEFONE"
```

Seguindo a programação do exemplo anterior da caderneta, a partir da linha 300, colocamos o setor de entrada de dados. Vamos definir a variável J para contar o número de campos em cada registro, e que será incrementada por uma alça FOR...NEXT:

```
300 LET NUM = NUM + 1
310 IF NUM > 25 THEN GOTO 900
320 CLS
330 PRINT "ENDERECO NO.";NUM
335 PRINT "==============="
340 FOR J = 1 TO 3
350 PRINT CAMPOS$(J);":";
355 INPUT DADOS$(NUM,J)
365 NEXT J
370 GOTO 40
```

Compare com o programa anterior, para verificar como este ficou muito mais compacto.

A seguir, programamos o segmento de modificação de dados. Inicialmente mostramos no alto da tela como estão os dados da pessoa solicitada, e depois pedimos os dados novos. Aqui também usamos alças FOR... NEXT:

```
400 CLS
410 PRINT "ENTRE O NUMERO DO
    REGISTRO A SER
    MODIFICADO:"
415 INPUT X
420 IF X < 1 THEN GOTO 415
425 IF X > NUM THEN GOTO 415
430 PRINT
435 PRINT "ENDERECO NO.";X
440 PRINT "==============="
445 FOR J = 1 TO 3
450 PRINT CAMPOS$(J);":";
    DADOS$(X,J)
455 NEXT J
460 PRINT
465 FOR J = 1 TO 3
470 PRINT "NOVO ";CAMPOS$
    (J);":";
475 INPUT DADOS$(X,J)
480 NEXT J
490 GOTO 40
```

Finalmente, podemos programar o segmento de procura.

```
500 CLS
510 PRINT "DIGITE O NOME DA
    PESSOA:";
520 INPUT X$
530 PRINT X$
540 FOR I = 1 TO NUM
550 IF X$ < > NOME$(I) THEN
    GOTO 580
555 FOR J = 2 TO 3
560 PRINT CAMPOS$(I,J)
565 NEXT J
570 PAUSE 600
575 GOTO 40
580 NEXT I
590 PRINT "NAO CONHECO ESTE
    NOME"
595 GOTO 510
```

Temos agora o nosso programa completo, com os comentários nos lugares apropriados:

```
10 REM ********************************
15 REM * PROGRAMA AGENDA 1.00          *
20 REM * PARA LOGICA SINCLAIR          *
25 REM * PROGRAMA R. SABBATINI         *
30 REM ********************************
32 REM
34 LET NUM = 0
35 DIM DADOS$(50,3,25)
36 DIM CAMPOS$(3)
37 LET CAMPOS$(1) = "NOME"
38 LET CAMPOS$(2) = "ENDERECO"
39 LET CAMPOS$(3) = "TELEFONE"
40 CLS
50 PRINT "              AGENDA TK"
60 PRINT " CADERNETA DE ENDERECOS"
70 PRINT
80 PRINT "              COMANDOS:"
90 PRINT "=========================="
100 PRINT "1            INCLUIR UM NOME"
120 PRINT "2            MODIFICAR DADOS"
130 PRINT "3            INFORMAR DADOS"
150 PRINT "4            FIM"
160 PRINT "=========================="
170 PRINT
172 REM
174 REM **** SOLICITAR E TESTAR OPCAO ****
176 REM
180 PRINT "DIGITE UM NUMERO:"
190 INPUT OPCAO
200 IF OPCAO < 1 THEN GOTO 190
210 IF OPCAO > 4 THEN GOTO 190
212 REM
214 REM **** DIMENSIONAR CONJUNTOS ****
216 REM
242 REM
244 REM **** DESVIAR PARA OPCAO ****
246 REM
250 IF OPCAO = 1 THEN GOTO 300
270 IF OPCAO = 2 THEN GOTO 400
280 IF OPCAO = 3 THEN GOTO 500
```

```
290 IF OPCAO = 4 THEN STOP
292 REM
294 REM **** ADICIONAR NOMES ****
296 REM
300 LET NUM = NUM + 1
310 IF NUM > 25 THEN GOTO 900
320 CLS
330 PRINT "ENDERECO NO.";NUM
335 PRINT "==============="
340 FOR J = 1 TO 3
350 PRINT CAMPOS$(J);":";
355 INPUT DADOS$(NUM,J)
360 PRINT "TELEFONE:"
365 NEXT J
370 GOTO 40
380 REM
382 REM **** MODIFICAR DADOS ****
384 REM
400 CLS
410 PRINT "ENTRE O NUMERO DO REGISTRO
    A SER MODIFICADO:"
415 INPUT X
420 IF X < 1 THEN GOTO 415
425 IF X > NUM THEN GOTO 415
430 PRINT
435 PRINT "ENDERECO NO.";X
440 PRINT "==============="
445 FOR J = 1 TO 3
450 PRINT CAMPOS$(J); ":"; DADOS$(X,J)
455 NEXT J
460 PRINT
465 FOR J = 1 TO 3
470 PRINT "NOVO"; CAMPOS$ (J);":";
475 INPUT DADOS$(X,J)
480 NEXT J
490 GOTO 40
492 REM
494 REM **** PROCURA DADOS ****
496 REM
500 CLS
510 PRINT "DIGITE O NOME DA PESSOA:";
520 INPUT X$
530 PRINT X$
540 FOR I = 1 TO NUM
550 IF X$ < > NOME$(I) THEN GOTO 580
555 FOR J = 2 TO 3
560 PRINT CAMPOS$(I,J)
565 NEXT J
570 PAUSE 600
575 GOTO 40
580 NEXT I
590 PRINT "NAO CONHECO ESTE NOME"
595 GOTO 510
```

Digite este programa no computador e teste-o com vários dados.

O mais interessante do programa é que ele é bastante amplo, pois se quiséssemos adicionar outros campos de informação (por exemplo, cidade, CEP, data de aniversário etc.), bastaria modificar o dimensionamento da linha 22Ø, adicionar os nomes no conjunto CAMPOS$ e, em seguida, modificar o valor final das alças FOR... NEXT, nas linhas 34Ø, 445, 465 e 555.

TESTE

10.1. Defina:

conjunto numérico: ______________________________

conjunto alfanumérico: ______________________________

vetor: ______________________________

matriz: ______________________________

índice: ______________________________

10.2. Qual é a função da declaração DIM ?

____

10.3. Escreva um programa para realizar a seguinte tarefa: um clube de aficionados de microcomputadores deseja fazer uma lista de todos os seus sócios, na qual conste a idade de cada um e a marca do microcomputador que possui. O programa, depois de entrados os dados, informará:

— o número de sócios que têm microcomputadores marca TK 85;
— o número de sócios que têm microcomputadores marca Apple;
— o número de sócios que têm outros tipos de microcomputador;
— a idade média dos sócios do clube;
— o nome do sócio mais novo do clube.

Utilize conjuntos unidimensionais chamados NOME$, IDADE e COMPUTADORE$ para registrar os dados acima.

10.4. Seja a matriz abaixo:

DIM DADOS(2,3)

	0	1	2	3
0				
1				
2				

Preencha as caselas da matriz com os dados gerados sucessivamente pelos programas abaixo:

a.
```
10 LET I=0
20 LET J=2
30 FOR M=I TO J
40 LET DADOS(M,M)=1
50 NEXT M
60 LET DADOS(2,3)=1
```

b.
```
70 FOR I=0 TO 2
80 FOR J=0 TO 2
90 IF I=J THEN GOTO 110
100 LET DADOS(I,J)=0
110 NEXT J
120 NEXT I
```

c.
```
130 LET SOMA=0
140 FOR COL=0 TO 3
150 LET SOMA=SOMA+DADOS(0,COL)
160 NEXT COL
170 PRINT SOMA
```

Que valor será mostrado na tela quando o computador executar a linha 170?

10.5. Escreva por extenso a capacidade, em número de linhas, de colunas e de caracteres, dos seguintes conjuntos literais:

a. DIM LIVROS$(100,50)

b. DIM X$(12,25,5)

c. DIM ARR$(1,100)

ESENHANDO COM O COMPUTADOR

Você já viu, no capítulo 5 (páginas 23 a 25), que existem alguns caracteres gráficos que podem ser usados dentro de comandos PRINT para fazer desenhos na tela.

Assim, por exemplo, podemos desenhar um belo padrão de blocos gráficos na tela utilizando o seguinte programa:

```
10 LET P$ = "■□▚▞■"
20 PRINT P$;
30 GOTO 20
```

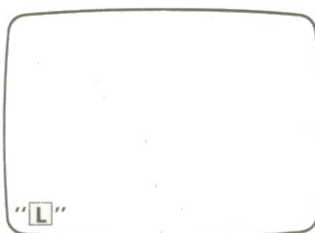
Observe que a linha de caracteres gráficos armazenados em P$ é formada pelos seguintes caracteres gráficos:

Você já aprendeu como são digitados estes caracteres gráficos no programa, utilizando a tecla GRAPHICS.

Quando você executar este programinha simples, com o comando RUN, aparecerá um padrão gráfico repetido, que encherá toda a tela, de cima para baixo.

Para fazer experiências com diferentes linhas de conjuntos gráficos, modifique o programa para aceitar os caracteres introduzidos pelo comando INPUT, assim:

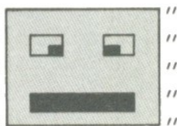
```
10 INPUT P$
20 PRINT P$;
30 GOTO 20
```

Quando o programa é executado pela primeira vez, aparece na tela o sinal de prontidão, que avisa ao *usuário que ele deve digitar uma linha de literais (caracteres):*

Nesse ponto, você deve pressionar as teclas SHIFT e 9, simultaneamente, para entrar no modo gráfico, e em seguida digitar de cinco a nove caracteres gráficos quaisquer.

Para dar seqüência ao programa, saia do modo gráfico, pressionando novamente as teclas SHIFT e 9, e depois a tecla NEW LINE. O mesmo efeito pode ser conseguido se você pressionar duas vezes a tecla NEW LINE.

Usando a sua imaginação, você poderá "inventar" muitos gráficos diferentes, fazendo uso de várias instruções PRINT.

Por exemplo, vamos fazer um programa para desenhar a cara do ET (o simpático personagem extraterrestre do filme do mesmo nome):

```
10 PRINT
20 PRINT "        "
30 PRINT "        "
40 PRINT "        "
50 PRINT "        "
60 PRINT "        "
```

Para fazermos o ET "espichar" o pescoço, podemos colocar os seguintes comandos, dentro de uma alça FOR:

```
70 FOR I = 1 TO 10
80 PRINT  "■■■"
90 NEXT I
```

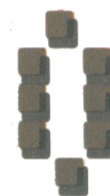
Veja o efeito interessante causado por este programa.

EXERCÍCIOS

1. Faça programas para realizar os seguintes desenhos na tela, usando uma seqüência de comandos PRINT gráficos:

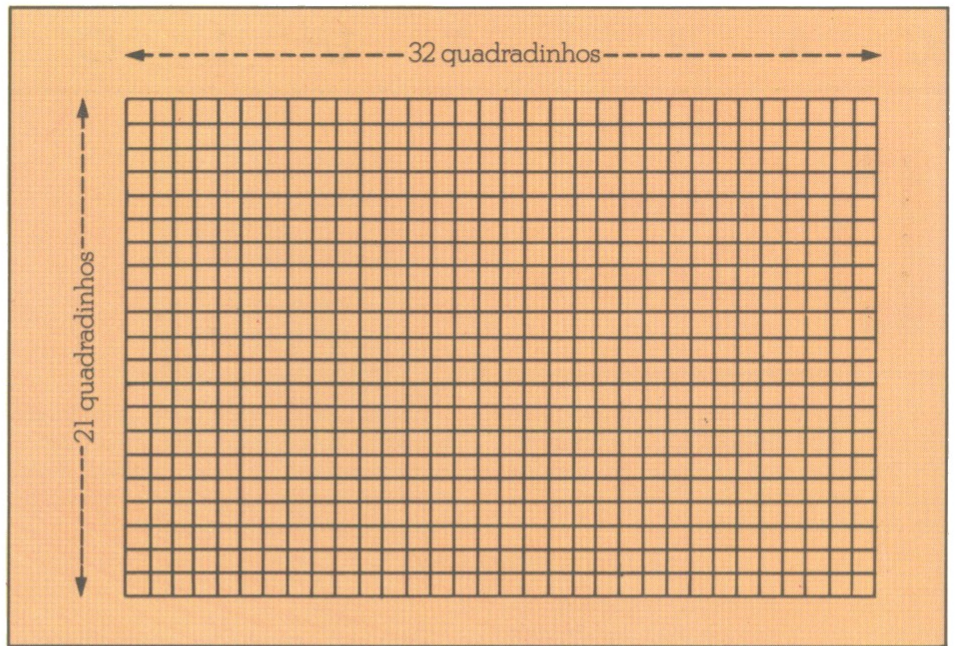
a. o mapa do Brasil, indicando por uma cruz a cidade em que você mora;
b. a bandeira do Brasil;
c. uma espaçonave flutuando acima de uma paisagem lunar;
d. uma locomotiva com dois vagões;
e. os arranha-céus de uma cidade (paisagem noturna).

2. Modifique o programa de treinamento de tabuada (páginas 71 a 72), para mostrar a figura de um palhacinho rindo, se a criança acertar o problema, e chorando, se a criança errar.

3. Invente outros desenhos e faça programas para eles.

utros truques gráficos

Uma das coisas mais fascinantes que podemos fazer com um computador pessoal é um videogame com desenhos que se movem (por exemplo, um avião que atravessa a tela de um lado para o outro). Esse tipo de técnica de programação é chamado de *animação gráfica*, pois ela corresponde à maneira como são produzidos em filmes os desenhos animados.

Para aprender a realizar uma animação gráfica no computador, primeiro temos que conhecer mais algumas características de programação da tela de vídeo.

A tela de vídeo de computadores do tipo TK 82, TK 83, TK 85 etc. pode ser comparada a uma folha de papel quadriculado, onde cada quadradi*nho corresponderia a uma posição determinada na tela:*

Cada posição na tela corresponde a um caractere.

Quando a tela é apagada por um comando CLS, ou logo após termos digitado o comando RUN, todas essas posições estarão em branco.

Ao encontrar um comando PRINT, as posições na tela serão ocupadas pelos caracteres enviados por esse comando.

Por exemplo, o programa:

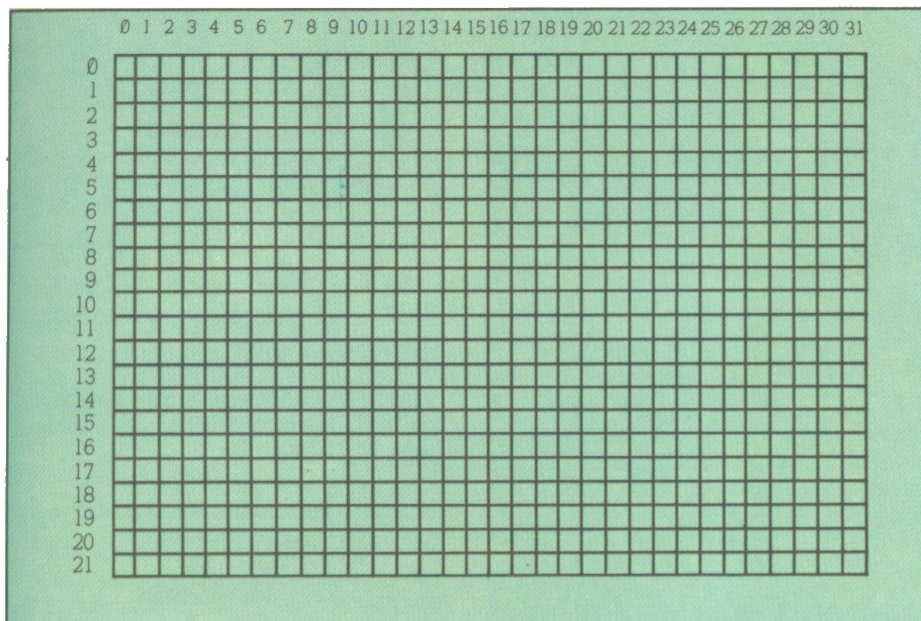
```
1Ø CLS
2Ø PRINT
3Ø PRINT "ALO"
4Ø PRINT TAB (5);"COMO VAI?"
5Ø STOP
```

vai escrever nas seguintes posições da tela:

Observe que cada comando PRINT faz o processo de escrita deslocar-se para a linha seguinte. A função TAB serve, no exemplo, para avançar dez posições, na segunda linha de impressão.

Podemos comparar o esquema de posições na tela a uma matriz ou tabela bidimensional, que você já aprendeu no capítulo anterior.

As linhas da tabela correspondem às linhas sucessivas na tela. Elas são numeradas de Ø a 21, de cima para baixo, começando pela linha superior da tela.

As colunas da tabela correspondem às colunas de impressão em uma determinada linha. Elas são numeradas de Ø a 31, da esquerda para a direita.

Na realidade, a tela de vídeo do seu computador tem 23 linhas. Mas as linhas 22 e 23 normalmente não podem ser usadas por um comando PRINT. Elas estão reservadas pelo interpretador BASIC para as seguintes funções:

linha 22: linha de entrada de dados (onde aparece o cursor "L" cada vez que um comando INPUT é encontrado no programa;
linha 23: é usada para mostrar mensagens de erro (por exemplo, 6/160).

Da mesma forma que em uma matriz, podemos *referenciar* cada posição da tela de vídeo, por meio dos seus índices, ou seja:

posição da linha

posição da coluna

O comando que faz esta referência é chamado de AT, e é sempre usado conjuntamente com o comando PRINT.

Por exemplo:

PRINT AT 5,10; "ALO" — escreverá a palavra ALO na linha nº 5, contando de cima para baixo, e começando na coluna 10 desta linha, contando da esquerda para a direita;

PRINT AT 20,0; "▄" — colocará na linha 20, e na coluna 0, o caractere gráfico ▄.

AT, em inglês, significa EM.

Portanto, PRINT AT 2,5 tem o seguinte significado:

ESCREVA EM 2,5

O primeiro valor que vem depois da palavra AT é o número da linha. O segundo valor (que é separado do primeiro por uma vírgula) é o número da coluna.

Esses números não podem ser menores do que zero nem maiores do que o número máximo de linhas ou colunas na tela, conforme o caso.

Por exemplo,

```
PRINT AT -2,20
PRINT AT 2,56
```

estão errados, pois, no primeiro exemplo, -2 é um número de linha que não existe e, no segundo, 56 é um número de coluna que também não existe.

Portanto, o comando PRINT AT é muito útil, pois permite que escrevamos qualquer coisa (caracteres normais ou gráficos), em qualquer ponto da tela, como um desenho em uma folha de papel.

Por exemplo, o programa:

```
10 PRINT AT 11,14;"MEIO"
```

escreverá no meio da tela, aproximadamente, a palavra MEIO:

Dessa forma, o programa não precisa ter várias linhas PRINT, colocadas em uma certa ordem.

Você vai entender melhor esse processo, ao estudar o programa abaixo.

O seu objetivo é escrever os nomes dos quatro pontos cardeais ("rosa dos ventos" ou bússola), nas quatro bordas da tela, assim:

A primeira coisa que você deve fazer é calcular as posições de linha e coluna de cada palavra.

Uma forma de fazer este cálculo é escrever as palavras em um papel quadriculado comum, marcado com os limites da tela.

No exemplo acima, teremos:

palavra	nº de linha	nº de coluna
NORTE	0	13
OESTE	11	0
LESTE	11	27
SUL	21	14

Escolhemos a coluna 13 para a palavra NORTE porque a coluna do meio é de número 15, e a palavra tem 5 letras. Assim, ela ficará aproximadamente centralizada na linha nº 0. Da mesma forma, escolhemos a coluna 14 para a palavra SUL.

As palavras OESTE e LESTE serão colocadas na linha do meio. Escolhemos 27 para o nº de coluna da palavra LESTE porque assim ela ocupará

até a coluna nº 31, ficando encostada na borda direita da tela.

Agora que o trabalho de cálculo está feito, escrever o programa é elementar:

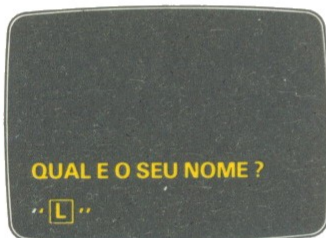
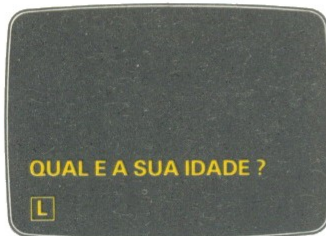
```
10 CLS
20 PRINT AT 0,13;"NORTE"
30 PRINT AT 11,0;"OESTE"
40 PRINT AT 11,27;"LESTE"
50 PRINT AT 21,14;"SUL"
```

Note que não faria diferença, para o resultado final, se tivéssemos escrito o programa assim:

```
10 CLS
20 PRINT AT 0,13;"NORTE"
30 PRINT AT 11,27;"LESTE"
40 PRINT AT 11,0;"OESTE"
50 PRINT AT 21,14;"SUL"
```

A única coisa que iria mudar seria a ordem em que o computador escreveria as palavras.

O comando PRINT AT tem outra aplicação muito interessante, para dotar seus programas de um aspecto mais "profissional".

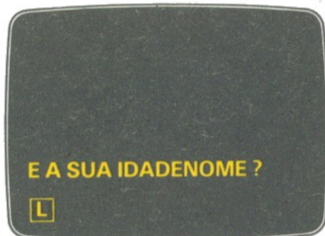
Ao solicitarmos alguma entrada ao operador de um programa, seria ideal que a mensagem aparecesse logo acima da linha de digitação (linha nº 22):

Depois de digitar o nome, queremos outra mensagem assim:

Para fazer isso, nada mais fácil:

```
10 PRINT AT 21,0;"QUAL E O SEU
   NOME ?"
20 INPUT NOME$
30 PRINT AT 21,0;"QUAL E A SUA
   IDADE ?"
40 INPUT IDADE
```

Note que a segunda mensagem (linha 30) é mais longa que a primeira (linha 10). Por isso, ao ser colocada a segunda mensagem na tela, ela encobrirá a mensagem anterior, lá colocada.

Entretanto, se ocorresse o contrário, isto é, se a segunda mensagem fosse mais curta que a primeira, as duas se "misturariam" (um pedaço da primeira continuaria na tela). Por exemplo, o programa

```
10 PRINT AT 21,0;"QUAL E O SEU
   NOME ?"
20 INPUT NOME$
30 PRINT AT 21,0;"E A SUA
   IDADE ?"
40 INPUT IDADE
```

originaria a seguinte mensagem, quando executada a linha 30:

Para evitar isso, temos antes que "limpar" a linha assim:

```
25 PRINT AT 21,0;"
```

o que é mais rápido do que dar um comando CLS (que iria limpar toda a tela, inclusive a linha 21).

Ou, então, colocar alguns espaços em branco após a segunda mensagem, de modo a superpô-los ao espaço ocupado pelo restante da mensagem anterior:

```
10 PRINT AT 21,0;"QUAL E O SEU
   NOME ?"
20 INPUT NOME$
30 PRINT AT 21,0;"E A SUA
   IDADE ?    "
40 INPUT IDADE
```

Um desenho animado

Vamos usar agora o comando PRINT AT para fazer uma animação gráfica.

Movimentaremos um aviãozinho na tela, fazendo-o voar da esquerda para a direita.

A imagem do avião será obtida por meio de uma linha assim:

PRINT " "

Vamos escolher uma linha da tela, por onde voará o avião. Por exemplo, a linha 10.

Para colocar o avião em posições sucessivas na tela, vamos utilizar uma alça FOR...NEXT, assim:

```
10 FOR J = 0 TO 26
20 PRINT AT 10,J;"       "
30 PRINT AT 10,J;"       "
40 NEXT J
```

A linha 20 serve para desenhar a imagem do avião, começando na linha 10, coluna J.

Assim, como J fica igual a 0, 1, 2, 3 .... etc., a cada repetição da alça FOR, que vai de 10 a 40, a imagem do avião será escrita nas posições 0, 1, 2, 3 ... etc. da linha 10, dando a impressão de locomoção.

A linha 30 serve para apagar o avião da posição J, antes de colocá-lo em nova posição. Por isso, são escritos cinco caracteres em branco, sobre a mesma posição anterior. Experimente não colocar esta linha e veja o que acontece.

O limite máximo da alça J (ou seja, 26) foi definido assim, pois o avião tem cinco caracteres.

Ao executar o programa, você pode observar que a imagem do aviãozinho dá a impressão de se movimentar com rapidez de um lado para outro da tela.

O programa, do jeito que está, já dá o máximo de velocidade aparente ao avião, se quisermos deslocá-lo uma posição de cada vez.

Para aumentar a velocidade de animação na tela, portanto, basta incrementarmos a posição do avião de 2 em 2 ou de 3 em 3, assim:

```
10 FOR J = 0 TO 26 STEP 2
20 PRINT AT 10,J;"            "
30 PRINT AT 10,J;"            "
40 NEXT J
```

Não podemos usar um número muito grande no STEP, entretanto, pois o avião "pulará" muito na tela, e perderemos o efeito contínuo de animação.

Para diminuir a velocidade do avião a operação é mais fácil: basta colocarmos uma alça FOR de retardo de tempo, entre as linhas 20 e 30 e as linhas 30 e 40:

```
10 FOR J = 0 TO 26
20 PRINT AT 10,J;"            "
25 FOR N = 1 TO 5
27 NEXT N
30 PRINT AT 10,J;"            "
35 FOR N = 1 TO 5
37 NEXT N
40 NEXT J
```

Se quisermos fazer um programa mais abrangente, que nos permita entrar dados de velocidade e linha de deslocamento do avião, teremos:

```
 1 CLS
 2 PRINT "O VOO DO AVIAO"
 4 PRINT AT 21,0;"ALTURA
   (0 A 20) :"
 6 INPUT ALT
 7 PRINT AT 21,0;"VELOCIDADE
   (0 A 20) :"
 8 INPUT VEL
 9 CLS
10 FOR J = 0 TO 26
20 PRINT AT 21-ALT,J;"            "
25 FOR N = 1 TO 20 - VEL
27 NEXT N
30 PRINT AT 21-ALT,J;"            "
35 FOR N = 1 TO 20 - VEL
37 NEXT N
40 NEXT J
50 GOTO 1
```

Note que velocidade e altura são respectivamente o inverso de retardo e posição da linha na tela. Por isso, subtraímos os valores entrados de seus valores máximos, para podermos inverter!

EXERCÍCIOS

1. Faça um programa de animação gráfica para fazer um foguetinho desenhado na parte de baixo da tela subir em trajetória vertical até o topo da mesma.

Dica: é semelhante ao programa do avião, apresentado anteriormente, só que o deslocamento é ao longo de uma coluna da tela (o número da linha na declaração AT é que muda, diminuindo à medida que o foguete sobe).

2. O que faz o seguinte programa na tela?

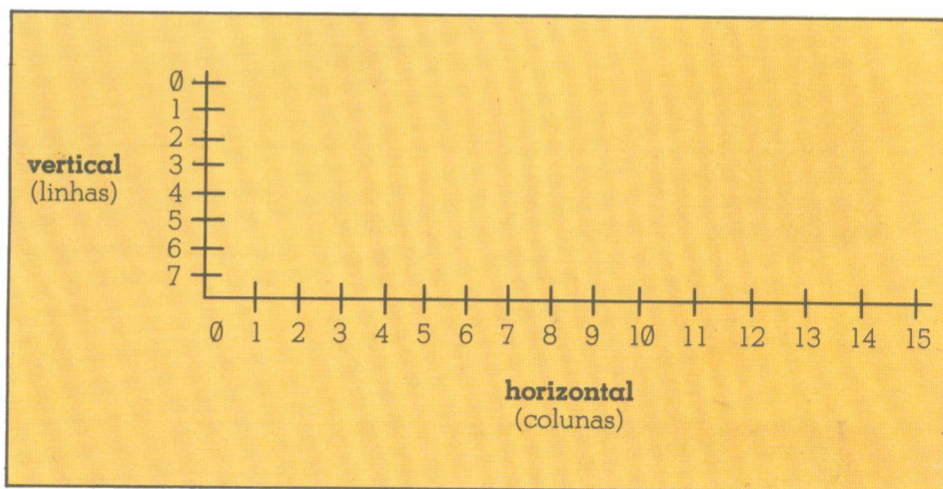
```
10 RAND
20 LET X=INT(RND*30)+1
30 LET Y=INT(RND*20)+1
40 PRINT AT Y,X;"*"
50 GOTO 20
```

Coloque o programa acima em seu computador e veja o resultado!

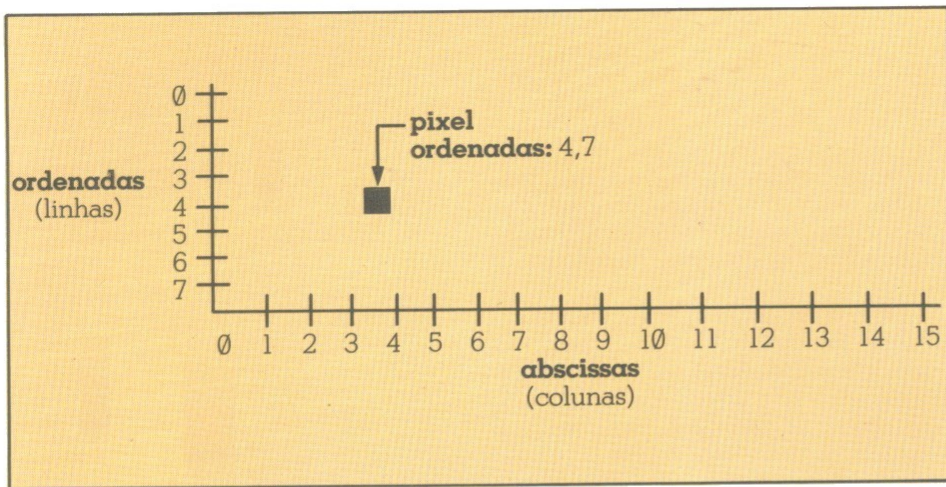
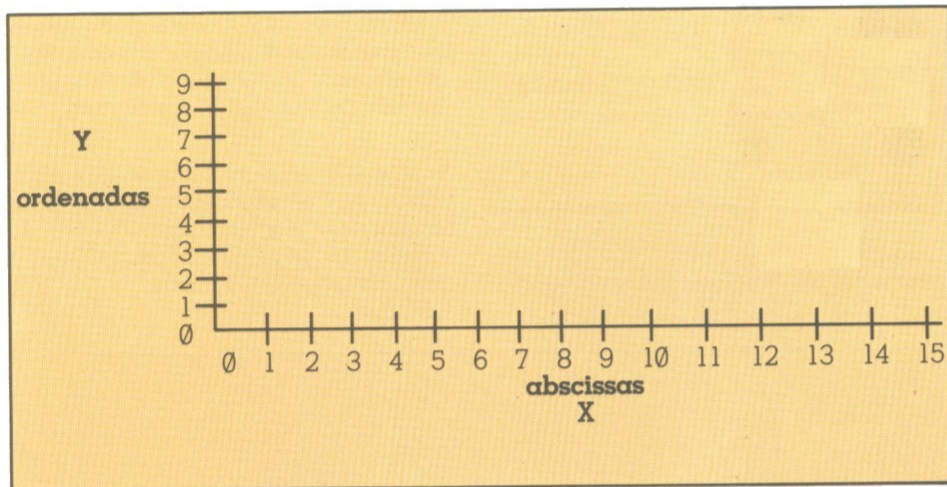
3. Modifique o programa acima para colocar um máximo de 100 asteriscos na tela.

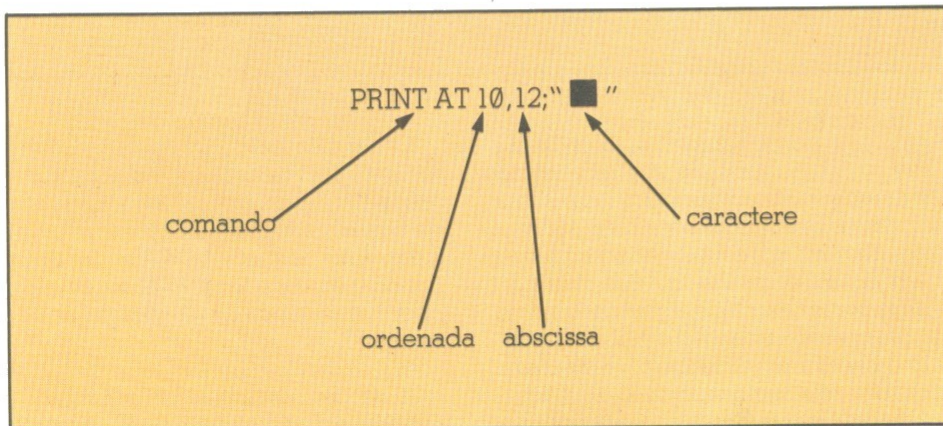
4. Faça um programa para desenhar uma torneira na parte superior da tela, e que fique pingando gotinhas de água continuamente (animação gráfica da gota caindo).

O sistema de localização de caracteres na tela de vídeo, usando o comando PRINT AT, como você percebeu muito bem, corresponde a uma espécie de mapa ou gráfico com posições horizontais e verticais.

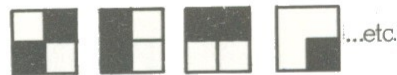
Esse sistema de referência é chamado de *sistema de coordenadas ortogonais*, pois os dois "eixos" de referência (o vertical e o horizontal) são perpendiculares, ou ortogonais entre si:

Um ponto qualquer no meio desta "grade" ou quadriculado é localizado através de suas *coordenadas*, ou seja, os números correspondentes à linha (coordenada vertical) e à coluna (coordenada horizontal) onde está. Para cada par de números desse tipo, temos apenas um ponto correspondente na tela. O contrário também é verdadeiro: para cada ponto na tela, temos apenas um determinado par de coordenadas, e apenas um.

O ponto na tela (um ponto nesse sistema de coordenadas no computador) recebe o nome de *pixel*. Este termo vem do inglês, e significa *picture element* (elemento gráfico). As coordenadas, por sua vez, recebem os nomes de *ordenada* (na vertical) e *abscissa* (na horizontal).

Já se convencionou chamar de X a abscissa e de Y a ordenada de um ponto:

Para quem já conhece o sistema de coordenadas ortogonais (também chamada de *cartesianas*, em homenagem ao seu inventor, o matemático e filósofo francês *René Descartes*), pode parecer estranho que o zero do eixo vertical (ordenadas) seja colocado na parte de cima da tela. Normalmente, nos livros de matemática ou de geometria, isto aparece assim:

Acontece que, por razões de facilidade de programação do controle da tela gráfica, como veremos, os idealizadores desse tipo de microcomputador preferiram um arranjo inverso, que às vezes nos atrapalha, mas não chega a prejudicar.

No sistema que vimos até agora (comando PRINT AT), podemos colocar um caractere qualquer (gráfico ou não gráfico), em qualquer ponto da tela.

Esse sistema nos permite colocar facilmente um texto ou um desenho em qualquer ponto da tela.

Entretanto, ele tem uma desvantagem: se quisermos fazer um gráfico mais detalhado, de modo a colocar na tela pontinhos menores, não dá.

Examinando os caracteres gráficos existentes no seu computador da linha Sinclair, você verá que eles são formados pela combinação de quatro subdivisões de um quadradinho gráfico:

Assim, qualquer caractere gráfico (excetuando-se os que têm elementos de cor cinza) pode ser composto preenchendo-se quadradinhos menores, em qualquer combinação:

1	2
3	4

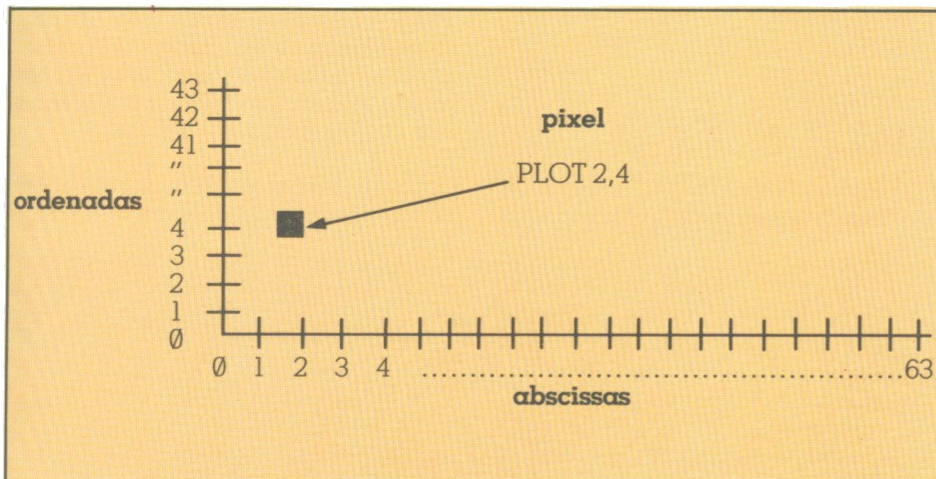
Portanto, podemos concluir que existe um *pixel* menor do que o caractere, e que corresponde a um tamanho quatro vezes menor, em área, do que o pixel que podemos manipular usando o comando PRINT AT.

A linguagem BASIC dos computadores da linha Sinclair tem comandos para manipular diretamente estes *pixels* menores. Com eles fica muito mais fácil traçar desenhos mais detalhados.

Esses comandos são:

PLOT - para colocar um pontinho na tela

UNPLOT - para apagar um pontinho da tela

O sistema de coordenadas dos comandos PLOT e UNPLOT consiste em 42 linhas, ou pontos no eixo vertical (ordenadas), e 64 colunas, ou pontos no eixo horizontal (abscissas):

Note que, nesse novo sistema de coordenadas, a posição do eixo vertical é mais "correta", ou seja, semelhante à convenção adotada pelo sistema cartesiano.

É muito fácil usar os comandos PLOT e UNPLOT. Basta fornecer apenas dois valores separados por uma vírgula, os quais indicam a ordenada e a abscissa do ponto desejado na tela:

PLOT 0,3 --› traça um ponto na abscissa = Ø e ordenada = 3

UNPLOT 5,2 --› apaga um ponto na abscissa = 5 e ordenada = 2

Os argumentos (valores X e Y) fornecidos aos comandos PLOT e UNPLOT podem ser de três tipos:

1. constantes numéricas	Exemplo:	PLOT 3,2
2. variáveis numéricas	Exemplos:	PLOT X,ABC PLOT II,X(2)
3. expressões matemáticas	Exemplos:	PLOT A + 1,B*24 + C PLOT X/2,SQR(X)

É claro que podemos misturar no mesmo PLOT qualquer um dos tipos acima:

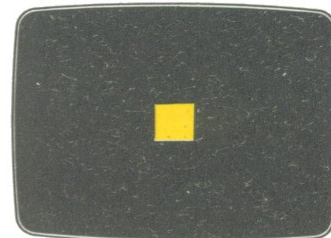
```
PLOT 2,BC*X
PLOT 12+C,K
etc.
```

fazendo pontinhos

Colocar e apagar pontos (pixels) na tela gráfica é bem fácil com a ajuda dos comandos PLOT e UNPLOT.

Experimente digitar, em modo direto (isto é, sem número de linha), o seguinte comando:

```
PLOT 31,21
```

Veja o que aconteceu na tela:

Apareceu um pixel bem no meio da tela de vídeo, ou seja, com as coordenadas X=31 e Y=21.

Agora experimente apagar este ponto com o comando:

```
UNPLOT 31,21
```

Veja como o pontinho se apagou!

Uma brincadeira interessante é repetir aqui aquele programa proposto nos exercícios da última lição, que serve para colocar pixels ao acaso na tela:

```
10 RANDOM
20 LET X=INT(RND*64)
30 LET Y=INT(RND*44)
40 PLOT X,Y
50 GOTO 20
```

Note que o número aleatório sorteado para a coordenada X varia entre Ø e 63 (a função RND nunca atinge 1.ØØØØ), e entre Ø e 43 para a coordenada Y. O efeito final se assemelha à chuva molhando uma calçada!

Outro programa interessante serve para desenharmos qualquer coisa na tela, fornecendo ao programa as coordenadas dos pontos que quisermos desenhar, um a um, na tela:

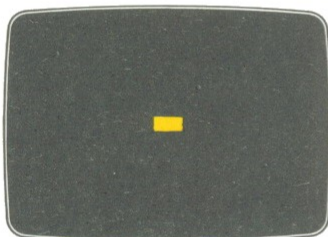
```
10 REM DESENHOS NA TELA
20 PRINT AT 21,0;"ENTRE VALOR
   DE X :"
30 INPUT X
40 PRINT AT 21,0;"ENTRE VALOR
   DE Y :"
50 INPUT Y
60 PLOT X,Y
70 GOTO 20
```

azendo retas

Retas paralelas aos eixos vertical e horizontal (ou seja, às bordas da tela) podem ser traçadas com grande facilidade. Basta dar vários comandos PLOT com coordenadas próximas umas às outras: se os pontinhos "grudarem" uns nos outros, darão a impressão visual de uma reta.

Por exemplo, digite sucessivamente os comandos diretos:

```
PLOT 31,21
PLOT 32,21
PLOT 33,21
```

E veja o resultado final:

Para não ficar repetindo muitas vezes os comandos PLOT, podemos usar os comandos FOR...NEXT para traçar uma reta:

```
10 FOR X=31 TO 33
20 PLOT X,21
30 NEXT X
```

Veja que o resultado final deste programa (após digitarmos o comando RUN) é idêntico ao mostrado acima com os comandos PLOT sucessivos. Isso acontece porque a variável X assume os valores 31, 32 e 33, controlados pela alça FOR, e o valor 21 da vertical é mantido constante.

Veja este programa para traçar uma reta na horizontal, começando em um determinado ponto e indo até outro:

```
10  REM RETAS HORIZONTAIS
15  REM
20  PRINT AT 21,0;"ENTRE VALOR
    INICIAL DE X :"
30  INPUT XI
40  PRINT AT 21,0;"ENTRE VALOR
    FINAL DE X :"
50  INPUT XF
60  PRINT AT 21,0;"EM QUE
    POSICAO HORIZONTAL:"
70  INPUT Y
80  FOR X=XI TO XF
90  PLOT X,Y
100 NEXT X
110 GOTO 20
```

Para entender melhor o esquema de coordenadas gráficas com os comandos PLOT e UNPLOT, vamos fazer um programa bem simples, que serve para traçar uma moldura ao redor da tela gráfica (ou seja, quatro traços finos colocados nos limites da tela gráfica).

Os quatro cantos da tela gráfica têm as seguintes coordenadas:

	X	Y
1. canto inferior esquerdo	0	0
2. canto inferior direito	63	0
3. canto superior esquerdo	0	43
4. canto superior direito	63	43

Portanto, para traçar quatro linhas de um canto a outro, fazemos:

```
10 REM MOLDURA PARA O TK
20 FOR I=0 TO 63
30 PLOT I,0
40 PLOT I,43
50 NEXT I
60 FOR J=0 TO 43
70 PLOT 0,J
80 PLOT 63,J
90 NEXT I
```

A primeira alça FOR...NEXT, que vai das linhas 20 até 50, traça simultaneamente as retas na parte de baixo (ordenada 0), e de cima (ordenada 43), respectivamente. A reta aparece na tela porque a variável I assume os valores 0, 1, 2, 3, 4... etc., até 63, "grudando" os pontinhos colocados na tela por cada instrução PLOT.

A segunda alça FOR...NEXT, que vai das linhas 60 até 90, faz a mesma coisa com as duas linhas verticais, situadas nas abscissas 0 e 63, respectivamente.

O uso da alça nos dá recursos bem poderosos para traçar retas de vários tipos. Por exemplo, para traçar uma reta pontilhada, assim:

....

basta "acender" um ponto sim, um ponto não, na tela, com a ajuda da instrução STEP

```
10 FOR I=0 TO 63 STEP 2
20 PLOT I,21
30 NEXT I
```

Percebeu?

EXERCÍCIOS

1. Modifique o programa que traça retas na horizontal, dados os pontos de início e fim, de modo que seja possível traçar retas na vertical.

2. Faça um programa para desenhar um retângulo qualquer na tela (apenas o seu contorno). O usuário deverá fornecer as coordenadas X e Y de cada canto da figura.

3. Faça um programa para desenhar um retângulo qualquer na tela. O usuário deverá fornecer os seguintes dados: coordenadas X e Y do canto inferior esquerdo do retângulo, tamanho do lado horizontal e tamanho do lado vertical.

4. Como o programa acima poderia ser modificado para preencher completamente o interior do retângulo (ou seja, desenhar um retângulo sólido)?

5. Faça um programa para traçar uma reta em diagonal, indo do canto inferior esquerdo ao canto superior direito da tela (atenção: este programa tem apenas três linhas!)

6. Modifique o programa que desenha pontinhos ao acaso na tela, para pedir ao usuário os limites inferior, superior, direito e esquerdo de um retângulo na tela, dentro do qual serão jogadas os pontinhos (ou seja, eles não ocuparão a tela inteira, como no exemplo dado no texto).

Animação gráfica com PLOT

Podemos realizar animações gráficas com os comandos PLOT e UNPLOT, também.

Veja este programa para simular uma bolinha caindo da parte de cima até a parte de baixo da tela:

```
10 FOR Y=43 TO 0 STEP -1
20 PLOT 31,Y
30 FOR T=1 TO 10
40 NEXT T
50 UNPLOT 31,Y
60 NEXT Y
```

A alça que vai da linha 30 a 40 causa um retardo de tempo, para fazer o pontinho se deslocar para baixo mais devagar, permitindo ao usuário enxergar o ponto traçado antes de ele se apagar da tela. Se você modificar o valor 10 da linha 30, poderá alterar esta velocidade para mais ou para menos.

Mas atenção: o uso do PLOT e UNPLOT para animações gráficas só dá bons efeitos quando o desenho a ser traçado for bem pequeno (poucos pontos). Caso contrário, o desenho se fará muito lentamente, bem como a animação gráfica (movimento) de suas partes. Para desenhos mais complexos é mais adequado usar o comando PRINT AT, conjuntamente com os caracteres gráficos disponíveis no teclado, como vimos na seção anterior.

RESPOSTAS AOS EXERCÍCIOS ANTERIORES

Exercício 2, pág. 80. Modifique o programa de cálculo dos pontos ganhos dos times, de modo que mostre na tela de resultados a diferença entre o número de pontos ganhos por cada time e a média geral.

Solução:

```
10 PRINT "CAMPEONATO BRASILEIRO DE FUTEBOL"
20 PRINT "       TABELA DE PONTOS GANHOS"
25 PRINT "                 VERSAO 3"
30 PRINT
40 REM
50 REM === ENTRADA DE DADOS
51 REM
52 PRINT "QUANTOS TIMES?"
54 INPUT NT
70 PRINT "ENTRE O NUMERO DE PONTOS GANHOS"
75 PRINT "PARA CADA TIME DA TABELA:"
80 PRINT
85 FOR N=1 TO NT
90 PRINT "TIME NUMERO  "; N; " ? ";
100 INPUT T(N)
110 PRINT T (N)
120 NEXT N
130 REM
140 REM === CALCULO DA MEDIA
150 REM
160 LET SOMA=0
165 CLS
170 FOR N=1 TO NT
180 LET SOMA=SOMA + T (N)
190 NEXT N
200 REM
210 REM === MOSTRA RESULTADOS
220 REM
400 LET MEDIA = SOMA/NT
410 PRINT
420 PRINT "MEDIA DOS PONTOS GANHOS = "; MEDIA
422 PRINT
425 FOR N=1 TO NT
426 PRINT "TIME NUMERO  "; N;" = "; T (N); "PONTOS GANHOS.  ";
428 PRINT "DIFERENCA DA MEDIA = "; MEDIA-T(N); "  PONTOS."
429 NEXT I
430 STOP
```

Exercício 2, pág. 88. Seja um conjunto DADO, com cinco linhas e quatro colunas. Escreva um programa com a alça FOR NEXT, para preencher com dados os elementos do conjunto DADO (por meio de um comando INPUT).

Solução:

```
10 FOR LIN=1 TO 5
20 FOR COL=1 TO 4
30 PRINT "ENTRE VALOR DA LI-
   NHA ";LIN;" E COLUNA ";COL;"
   = ";
40 INPUT DADO (LIN, COL)
50 PRINT DADO (LIN, COL)
60 NEXT COL
70 NEXT LIN
```

Exercício 3, pág. 88. Escreva um programa para preencher com zeros todos os elementos do conjunto DADO.

Solução:

```
10 FOR LIN=1 TO 5
20 FOR COL=1 TO 4
30 LET DADO (LIN, COL)=0
40 NEXT COL
50 NEXT LIN
```

Exercício 5, pág. 88. Faça um programa para copiar uma das linhas da matriz DADO, para um vetor (conjunto unidimensional) chamado COPIA.

Solução:

```
10 PRINT "QUE LINHA (1 A 5) ?"
20 INPUT LIN
30 FOR COL=1 TO 4
40 LET COPIA (COL) =DADO
   (LIN, COL)
50 NEXT COL
```

Exercício 6, pág. 88. Faça um programa para copiar uma das colunas da matriz DADO, para um vetor chamado COPIA.

Solução:

```
10 PRINT "QUE COLUNA (1 A 4) ?"
20 INPUT COL
30 FOR LIN=1 TO 5
40 LET COPIA (LIN)=DADO
   (LIN, COL)
50 NEXT LIN
```

Exercício 7, pág. 88. Escreva um programa para preencher com o valor 1 todos os elementos da diagonal principal de um conjunto A(10, 10).

Solução:

```
10 DIM A(10, 10)
20 FOR I=1 TO 10
30 LET A(I,I)=1
40 NEXT I
```

Exercício 10, pág. 88. Modifique o programa completo de cálculo de boletim de notas, de modo a realizar as seguintes funções adicionais:

a. Calcular a média global da classe (todas as notas);

b. Indicar, ao lado da média de cada aluno, se ela está acima ou abaixo da média global da classe.

Solução:

Vamos fazer também a modificação necessária para resolver outro exercício proposto:

Exercício 2, pág. 87. Entrar e mostrar o nome de cada aluno.

```
10 REM ==============================
20 REM = BOLETIM DE NOTAS DA CLASSE
30 REM = BASIC PARA TK 85
40 REM = R. SABBATINI VERSAO 2
50 REM ==============================
60 REM
70 REM **** INICIALIZACAO ****
80 REM
90 DIM NOTA (50, 15)
95 DIM NO$ (50, 15)
100 CLS
110 PRINT "CALCULO DO BOLETIM DE NOTAS"
120 PRINT "============================"
130 REM
140 REM **** ENTRADA DE PARAMETROS ****
150 REM
160 PRINT
170 PRINT "NUMERO DE ALUNOS NA CLASSE ?"
180 INPUT NA
190 IF NA > 50 THEN GOTO 170
200 PRINT NA
```

```
210 PRINT "NUMERO DE MATERIAS  ?";
220 INPUT NM
230 IF NM > 15 THEN GOTO 210
240 PRINT NM
250 REM
260 REM **** ENTRADA DE DADOS ****
270 REM
280 PRINT "ENTRE AS NOTAS DOS ALUNOS:"
290 PRINT
300 FOR I=1 TO NA
310 PRINT "ALUNO NO. ";I
315 PRINT "NOME DO ALUNO: ";
317 INPUT NO$ (I)
320 FOR J=1 TO NM
330 PRINT TAB (7); "NOTA DA MATERIA "; J;" ?";
340 INPUT NOTA (I, J)
350 IF NOTA (I, J) <0 THEN GOTO 340
360 IF NOTA (I, J) >10 THEN GOTO 340
370 PRINT NOTA (I, J)
380 NEXT J
390 REM
400 REM **** MOSTRA NOTAS NA TELA ***
410 REM **** E CALCULA MEDIAS ***
420 REM
421 LET SOMA=0
422 FOR I=1 TO NA
424 FOR J=1 TO NM
426 LET SOMA=SOMA+NOTA (I, J)
427 NEXT J
428 NEXT I
429 LET MCLAS=SOMA/(I*J)
430 CLS
440 FOR I=1 TO NA
450 PRINT "ALUNO  ";I;" — "; NO$ (I);":"
460 LET SOMA=0
470 FOR J=1 TO NM
480 PRINT TAB(10+5*J); NOTA (I, J);
490 LET SOMA=SOMA+NOTA (I,J)
500 NEXT J
510 PRINT " MED "; SOMA/NM
515 IF SOMA/NM> =MCLAS THEN PRINT " *ACIMA"
517 IF SOMA/NM<MCLAS THEN PRINT " *ABAIXO"
520 NEXT I
530 REM
540 REM **** CALCULA MEDIA DAS MATERIAS ****
550 REM
560 FOR J=1 TO NM
570 LET SOMA=0
580 FOR I=1 TO NA
590 LET SOMA=SOMA+NOTA (I, J)
600 NEXT I
610 PRINT
620 LET MED=SOMA/NA
630 PRINT "MEDIA NA MATERIA ";J;" = "; MED
640 NEXT I
650 STOP
```

Exercício 1, pág. 92. Modifique o programa de emissão de nota de despesas de modo que tenha as seguintes funções adicionais:

a. Entrar a data de emissão da nota e o nome do cliente para mostrar na tela de emissão;

b. Armazenar o valor total de cada nota, cumulativamente em uma variável RENDA, e mostrar este valor se o nome do cliente for "FIM".

Solução:

```
10 CLS
20 PRINT "EMISSAO DE NOTA DE DESPESAS"
30 PRINT "CASA DE FRIOS "AZEITONA DOURADA" LTDA"
32 PRINT
34 PRINT "DIGITE A DATA DE HOJE:"
36 INPUT DAT$
40 PRINT
50 DIM M$ (5,20)
55 DIM TT (5)
60 LET M$(0) = "AZEITONAS PRETAS"
70 LET M$(1) = "AZEITONAS VERDES"
80 LET M$(2) = "PRESUNTO GORDO"
90 LET M$(3) = "PRESUNTO MAGRO"
100 LET M$(4) = "MOZARELA"
110 LET M$(5) = "QUEIJO PRATO"
115 LET RENDA = 0
120 REM
130 REM ***** ENTRADA DOS DADOS DA COMPRA *****
140 REM
142 PRINT "DIGITE O NOME DO CLIENTE:"
144 INPUT NO$
146 IF NO$ = "FIM" THEN GOTO 900
148 PRINT
150 PRINT "DIGITE A QUANTIDADE COMPRADA E DEPOIS"
160 PRINT "O PREÇO POR QUILOGRAMA PARA CADA MER-"
170 PRINT "CADORIA APRESENTADA NA TELA. DIGITE 0"
180 PRINT "SE O CLIENTE NAO COMPROU A MERCADORIA"
190 PRINT
200 FOR I = 1 TO 5
210 PRINT M$(I);" ? "
220 INPUT Q(I)
230 IF Q(I) = 0 THEN GOTO 250
240 INPUT PR(I)
250 NEXT I
260 REM
270 REM ***** IMPRESSAO DA NOTA *****
280 REM
290 CLS
300 PRINT "                NOTA DE DESPESAS"
305 PRINT
307 PRINT "CLIENTE: "; NO$;" DATA:";DAT$
310 PRINT
320 PRINT "MERCADORIA"; TAB(17); "QUANT PRECO"
325 LET TOT= 0
330 FOR I = 1 TO 5
340 IF Q(I) = 0 THEN GOTO 360
350 PRINT M$(I); TAB(17); Q(I); TAB(22); Q(I)*PR(I)
355 LET TOT=TOT+Q(I)*PR(I)
360 NEXT I
365 PRINT "TOTAL"; TAB(22); TOT
366 LET RENDA = RENDA + TOTAL
370 PAUSE 600
380 CLS
```

```
390 GOTO 150
800 REM
810 REM ***** MOSTRAR RENDA FINAL *******
820 REM
900 CLS
910 PRINT "RESULTADO CUMULATIVO = CR$ ";RENDA
920 STOP
```

Exercício 1, pág. 95. Modifique o programa AGENDA para incluir as seguintes funções adicionais:

1 - Exclusão de um nome da caderneta (*dica:* o programa solicita o número do registro a ser excluído e em seguida coloca espaços em branco nos conjuntos correspondentes);

2 - Listagem de todos os nomes colocados na caderneta (*dica:* coloque um registro em cada linha da tela, começando com o número do registro, depois o nome, o endereço e o telefone).

Solução (programa da pág. 94 completo):

```
10 REM ****************************
15 REM * PROGRAMA AGENDA 1.00  *
20 REM * PARA LOGICA SINCLAIR  *
25 REM * PROGRAMA: R. SABBATINI *
30 REM ****************************
32 REM
33 REM
34 REM ***** DIMENSIONAR CONJUNTOS *****
35 REM
36 DIM NOME$(50,25)
37 DIM ENDERECO$(50,25)
38 DIM TELEFONE$(50,10)
40 CLS
50 PRINT "          AGENDA TK"
60 PRINT "  CADERNETA DE ENDERECOS"
70 PRINT
80 PRINT "          COMANDOS:"
90 PRINT "============================"
100 PRINT "1      INCLUIR UM NOME"
120 PRINT "2      MODIFICAR DADOS"
130 PRINT "3      INFORMAR DADOS"
150 PRINT "4      FIM"
160 PRINT "============================"
170 PRINT
172 REM
174 REM ***** SOLICITAR E TESTAR OPCAO *****
176 REM
180 PRINT "DIGITE UM NUMERO:"
190 INPUT OPCAO
200 IF OPCAO <1 THEN GOTO 190
210 IF OPCAO >6 THEN GOTO 190
242 REM
244 REM ***** DESVIAR PARA OPCAO *****
246 REM
250 IF OPCAO=1 THEN GOTO 300
270 IF OPCAO=2 THEN GOTO 400
280 IF OPCAO=3 THEN GOTO 500
284 IF OPCAO=4 THEN GOTO 600
286 IF OPCAO=5 THEN GOTO 700
```

```
290 IF OPCAO=6 THEN STOP
292 REM
294 REM ***** ADICIONAR NOMES *****
296 REM
300 LET NUM=NUM+1
310 IF NUM>25 THEN GOTO 900
320 CLS
330 PRINT "ENDERECO NO.";NUM
335 PRINT "==============="
340 PRINT "NOME:"
345 INPUT NOME$(NUM)
350 PRINT "ENDERECO:"
355 INPUT ENDERECO$(NUM)
360 PRINT "TELEFONE:"
365 INPUT TELEFONE$(NUM)
370 GOTO 40
380 REM
382 REM ***** MODIFICAR DADOS *****
384 REM
400 CLS
410 PRINT "ENTRE O NUMERO DO REGISTRO A SER MODIFICADO:"
415 INPUT X
420 IF X<1 THEN GOTO 415
425 IF X>25 THEN GOTO 415
430 PRINT
435 PRINT "ENDERECO NO.";X
440 PRINT "==============="
445 PRINT "NOME:  ";NOME$(X)
450 PRINT "ENDERECO:  ";ENDERECO$(X)
455 PRINT "TELEFONE:  ";TELEFONE$(X)
460 PRINT "NOVO NOME:"
465 INPUT NOME$(X)
470 PRINT "NOVO ENDERECO:"
475 INPUT ENDERECO$(X)
480 PRINT "NOVO TELEFONE:"
485 INPUT TELEFONE$(X)
490 GOTO 40
492 REM
494 REM ***** PROCURA DADOS *****
496 REM
500 CLS
510 PRINT "DIGITE O NOME DA PESSOA:";
520 INPUT X$
530 PRINT X$
540 FOR I=1 TO NUM
550 IF X$<>NOME$(I) THEN GOTO 580
560 PRINT "ENDERECO:";ENDERECO$(I)
565 PRINT "TELEFONE:";TELEFONE$(I)
570 PAUSE 600
575 GOTO 40
580 NEXT I
590 PRINT "NAO CONHECO ESTE NOME"
595 GOTO 510
600 REM
610 REM ***** LISTA DADOS *****
620 REM
630 CLS
640 FOR I=1 TO NUM
650 PRINT NOME$(I);" ";ENDERECO$(I);" ";TELEFONE$(I)
660 NEXT I
670 PAUSE 600
680 GOTO 40
700 REM
710 REM ***** MODIFICAR DADOS *****
720 REM
730 CLS
740 PRINT "ENTRE O NUMERO DO REGISTRO A SER SUPRIMIDO:"
```

```
750 INPUT X
760 IF X<1 THEN GOTO 415
770 IF X>25 THEN GOTO 415
780 LET NOME$(X)=""
790 LET ENDERECO$(X)=""
800 LET TELEFONE$(X)=""
900 GOTO 40
```

Exercício 1, pág. 104. Faça um programa de animação gráfica para fazer um foguetinho desenhado na parte de baixo da tela subir em trajetória vertical até o topo da mesma.

Solução:

```
10 FOR L=22 TO 1 STEP -1
20 PRINT AT L,15;"▄█▄"
30 FOR J=1 TO 20
40 NEXT J
50 PRINT AT L,15;"   "
60 NEXT L
```

Exercício 3, pág. 104. Modifique o programa do Exercício 2 da página 104 para colocar um máximo de 100 asteriscos na tela.

Solução:

```
10 RANDOM
15 FOR I=1 TO 100
20 LET X=INT(RND*30)+1
30 LET Y=INT(RND*20)+1
40 PRINT AT Y,X;"*"
50 NEXT I
```

Exercício 4, pág. 104. Faça um programa para desenhar uma torneira na parte superior da tela, e que fique pingando gotinhas de água continuamente (animação gráfica da gota caindo).

Solução:

```
10 PRINT AT 1, 10;"███"
20 PRINT AT 2, 10;"█▀▀"
30 FOR I=3 TO 20
40 PRINT AT I,10;" ▀ "
50 FOR J=1 TO 20
60 NEXT J
70 PRINT AT I,10;"   "
80 NEXT I
90 GOTO 30
```

Exercício 1, pág. 107. Modifique o programa que traça retas na horizontal, dados os pontos de início e fim, de modo que seja possível traçar retas na vertical.

Solução:

```
10 REM RETAS VERTICAIS
15 REM
20 PRINT AT 21,0;"ENTRE
   VALOR INICIAL DE Y :"
30 INPUT YI
40 PRINT AT 21,0;"ENTRE
   VALOR
   FINAL DE Y :"
50 INPUT YF
60 PRINT AT 21,0;"EM QUE
   POSICAO VERTICAL :"
70 INPUT X
80 FOR Y=YI TO YF
90 PLOT X,Y
100 NEXT Y
110 GOTO 20
```

Exercício 2, pág. 107. Faça um programa para desenhar um retângulo qualquer na tela (apenas o seu contorno). O usuário deverá fornecer as coordenadas X e Y de cada canto da figura.

Solução:

```
10 REM RETANGULO PARA O
   TK
20 PRINT "ENTRE 2 POSI-
   COES PARA LADOS HORI-
   ZONTAIS:"
30 INPUT Y1
40 INPUT Y2
50 PRINT "ENTRE 2 POSI-
   COES PARA LADOS VER-
   TICAIS:"
60 INPUT X1
70 INPUT X2
80 CLS
90 FOR X=X1 TO X2
100 PLOT X,Y1
110 PLOT X,Y2
120 NEXT X
130 FOR Y=Y1 TO Y2
140 PLOT X1,Y
150 PLOT X2,Y
160 NEXT Y
```

Exercício 3, pág. 107. Faça um programa para desenhar um retângulo qualquer na tela. O usuário deverá fornecer os seguintes dados: coordenadas X e Y do canto inferior esquerdo do retângulo, tamanho do lado horizontal e tamanho do lado vertical.

Solução:

```
10 REM RETANGULO PARA O
   TK
12 PRINT "COORDENADA X
   DA ORIGEM:"
14 INPUT X1
16 PRINT "COORDENADA Y
   DA ORIGEM:"
18 INPUT Y1
20 PRINT "TAMANHO DOS
   LADOS HORIZONTAIS:"
30 INPUT H
50 PRINT "TAMANHO DOS
   LADOS VERTICAIS:
60 INPUT V
80 CLS
90 FOR X=X1 TO X1+H
100 PLOT X,Y1
110 PLOT X,Y1+V
120 NEXT X
130 FOR Y=Y1 TO Y1+V
140 PLOT X1,Y
150 PLOT X1+H,Y
160 NEXT Y
```

Exercício 4, pág. 107. Como o programa acima poderia ser modificado para preencher completamente o interior do retângulo (ou seja, desenhar um retângulo sólido)?

Solução:

```
10 REM RETANGULO PARA O
   TK
12 PRINT "COORDENADA X
   DA ORIGEM:"
14 INPUT X1
16 PRINT "COORDENADA Y
   DA ORIGEM:"
18 INPUT Y1
20 PRINT "TAMANHO DOS
   LADOS HORIZONTAIS:"
30 INPUT H
50 PRINT "TAMANHO DOS
   LADOS VERTICAIS:"
60 INPUT V
80 CLS
90 FOR X=X1 TO X1+H
130 FOR Y=Y1 TO Y1+V
140 PLOT X,Y
160 NEXT Y
170 NEXT X
```

Exercício 5, pág. 107. Faça um programa para traçar uma reta em diagonal, indo do canto inferior esquerdo ao canto superior direito da tela (atenção: este programa tem apenas três linhas!)

Solução:

```
10 FOR I=0 TO 40
20 PLOT I, I
30 NEXT I
```

Exercício 6, pág. 107. Modifique o programa que desenha pontinhos ao acaso na tela para pedir ao usuário os limites inferior, superior, direito e esquerdo de um retângulo na tela, dentro do qual serão jogados os pontinhos.

Solução:

```
10 RANDOM
15 FOR I=1 TO 100
20 LET X=INT(RND*30)+1
30 LET Y=INT(RND*20)+1
40 PRINT AT Y,X;"*"
50 NEXT I
```

RESPOSTAS AO TESTE ANTERIOR

Teste 3, pág. 98. Escreva um programa para realizar a seguinte tarefa: um clube de aficionados de microcomputadores deseja fazer uma lista de todos os seus sócios, na qual conste a idade de cada um e a marca do microcomputador que possui. O programa, depois de entrados os dados, informará:

- o número de sócios que têm microcomputadores marca TK-85;
- o número de sócios que têm microcomputadores marca Apple;
- o número de sócios que têm outros tipos de microcomputador;
- a idade média dos sócios do clube;
- o nome do sócio mais novo do clube.

Utilize conjuntos unidimensionais chamados NOME$, IDADE e COMPUTADOR$ para registrar os dados acima.

Solução:

```
10 REM ==================
20 REM CADASTRO DO CLUBE
30 REM PARA TK-85
40 REM ==================
50 REM
60 DIM NOME$(20,20)
70 DIM IDADE(20)
80 DIM COMPUTADOR$(20,12)
90 LET SOCIOS=0
100 REM
110 REM ***** ENTRADA DE DADOS *****
120 REM
130 PRINT "ENTRADA DOS DADOS DOS SOCIOS."
140 PRINT "PARA TERMINAR DIGITE *"
150 PRINT
160 CLS
170 PRINT "SOCIO NO.";SOCIOS+1
180 PRINT AT 21,0;"NOME DO SOCIO:"
190 INPUT NOME$(SOCIOS+1)
200 IF NOME$(SOCIOS+1)="*" THEN GOTO 300
210 LET SOCIOS=SOCIOS+1
220 PRINT "IDADE DO SOCIO:"
230 INPUT IDADE(SOCIOS)
240 PRINT "COMPUTADOR QUE TEM:"
250 INPUT COMPUTADOR$(SOCIOS)
260 GOTO 160
270 REM
280 REM ***** PESQUISAS *****
290 REM
300 LET TK85=0
310 LET APPLE=0
320 LET SOMA=0
330 LET MENOR=99
340 FOR I=1 TO SOCIOS
350 IF COMPUTADOR$(I)="TK-85" THEN LET TK85=TK85+1
360 IF COMPUTADOR$(I)="APPLE" THEN LET APPLE=APPLE+1
370 IF IDADE(I) (MENOR THEN LET MENOR=I
380 LET SOMA=SOMA+IDADE(I)
390 NEXT I
400 REM
410 REM ***** RESULTADOS *****
420 REM
430 PRINT "SOCIOS COM TK-85=";TK85
440 PRINT "SOCIOS COM APPLE=";APPLE
450 PRINT "OUTROS COMPUT. =";SOCIOS-APPLE-TK85
460 PRINT
470 PRINT "IDADE MEDIA=";SOMA/SOCIOS
480 PRINT "SOCIO CACULA=";NOME$(MENOR)
490 STOP
```

ÍNDICE REMISSIVO

A

B

C

D

E

F

G

I

J

L

M

N

O

P

SUMÁRIO

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

CAPÍTULO 6

CAPÍTULO 7

CAPÍTULO 8

CAPÍTULO 9